Computer persönlich

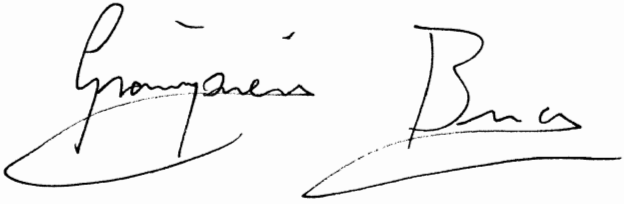
Computerspiele & Wissenswertes

Commodore 64

Nützliche Maschinenprogramme zum Eintippen
Tips · Grafik · Basic-Erweiterungen

Computerspiele und Wissenswertes Commodore 64

Nr. 271



Deutsche Übersetzung:
Peter Lüke

Computerspiele und Wissenswertes Commodore 64

Nützliche Maschinenprogramme zum Eintippen · Tips · Grafik · Basic-Erweiterungen

Markt & Technik Verlag

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Computerspiele und Wissenswertes Comodore 64 :
nützl. Maschinenprogramme zum Eintippen ; Tips ;
Grafik ; Basic-Erweiterungen / dt. Übers. Peter Lüke. —
Haar bei München : Markt u. Technik Verlag, 1984
(Computer persönlich)
Einheitssacht.: More on the sixtyfour <dt.>
ISBN 3-922120-62-8NE: Lüke, Peter [Übers.]; EST

Die Informationen im vorliegenden Buch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Die gewerbliche Nutzung der in diesem Buch gezeigten Modelle und Arbeiten ist nicht zulässig.

»Commodore« ist eine Produktbezeichnung der Commodore Büromaschinen GmbH, Frankfurt, die ebenso wie der Name »Commodore« Schutzrechte genießt. Der Gebrauch bzw. die Verwendung bedarf der Erlaubnis der Schutzrechtsinhaberin.

Autorisierte Übersetzung und Bearbeitung des im Verlag ELCOMP PUBLISHING, INC. erschienenen Buches: »More on the Sixtyfour 64«.

English edition © Copyright 1983
by ELCOMP PUBLISHING, INC., 53 Redrock Lane, Pomona, CA 91766, USA
All rights reserved

ISBN 3-922120-62-8

© 1984 by Markt & Technik, 8013 Haar bei München
Alle Rechte vorbehalten
Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Eimannsberger, München
Printed in Germany

Vorwort zur deutschen Ausgabe

Immer mehr Besitzer eines Commodore 64 schreiben Programme in Maschinensprache. Dabei tauchen immer wieder die gleichen Probleme auf. Für den einen Drucker braucht man eine Centronics-Schnittstelle, für einen anderen ein serielles RS-232-Interface. Häufig benötigt man die genaue Zeit oder man möchte gern den Befehlssatz des Basic erweitern. Ebenso oft lassen sich Projekte nur mit schneller Arithmetik realisieren. All dies sind Aufgabenstellungen, die man sinnvoll nur in Assemblersprache lösen kann.

Unabhängig davon, um welche Problematik es sich handelt, benutzen die meisten Programme für die Ein-/Ausgabe und für andere Zwecke immer wieder die gleichen oder zumindest ähnliche Programmsequenzen. Dieses Buch wendet sich an den fortgeschrittenen Programmierer, der lieber auf erprobte Standardlösungen zurückgreift, als jede Sequenz jedesmal neu zu entwickeln. Der Einsatz der fertigen Problemlösungen wird durch im Buch als Listing enthaltene Editor- und Assemblerprogramme unterstützt. Für alle diejenigen, denen das Eintippen zu mühsam ist, hält der Verlag eine Diskette mit allen im Buch gezeigten Programmen bereit.

Inhaltsverzeichnis

Vorwort zur deutschen Ausgabe 5

Inhaltsverzeichnis 7

1. Ein- und Ausgabe von Zahlen 9
   Hexadezimale Eingabe 11
   Hexadezimale Ausgabe 12
   Dezimale Eingabe 13
   Dezimale Ausgabe 15
   16 Bit Addition und Subtraktion 16
   16 Bit Multiplikation 17
   16 Bit Division 19

2. Textausgabe 21

3. Rekursion 25

4. Das Dateisystem 35
   Abspeichern und Laden von Programmen im OS 43

5. Die RS232-Schnittstelle 49

6. Die Echtzeituhr 55

7. Wie man neue BASIC-Befehle einfügt 69

8. Hochauflösende Grafik 73

9. Hilfsprogramm zur Diskettenbedienung 81

10. Wie man Gerätetreiber hinzufügt 89

11. Eine preiswerte Centronics-Schnittstelle 93

12. Formatierter Ausdruck 101

13. Ausgabe des Bildinhalts auf einen Drucker 105
    Bildschirmausdruck über die RS232-Schnittstelle 109

14. Terminal 113

15. Wie man den C-64 mit einem ATARI verbindet 127

16. Die RESTORE-Taste 133

17. Schnelle Ausgabe von Sedezimalzahlen 139

18. Einige Tips 145
    Wie man A/D-Wandler verwendet 147
    ROM ins RAM kopieren 147
    Groß- und Kleinschreibung auf dem 2022 148

Anhang Editor und Assembler 149
    Der Editor 151
    Der Assembler 153

1

Ein- und Ausgabe von Zahlen

Ein- und Ausgabe von Zahlen

Oft möchte man Zahlen direkt über den Bildschirm ein- und ausgeben. Die nachfolgenden Programme zeigen diese Möglichkeit mit sedezimalen Zahlen (auch hexadezimale Zahlen genannt) und mit dezimalen Zahlen.

Hexadezimale Eingabe

Dieses Programm ermöglicht die Eingabe hexadezimaler Zahlen mittels Tastatur. Die eingegebene Zahl wird auf dem Bildschirm angezeigt. Die Eingabe wird beendet, wenn eine Zahl außerhalb des hexadezimalen Zahlenbereichs (0...F) eingegeben wird.

Das Programm löscht als erstes die Speicherstellen EXPR und EXPR+1. So wird ein Ergebnis, nämlich 0, auf jeden Fall angezeigt, auch wenn eine ungültige Zahl eingegeben wurde. Als nächstes wird ein Zeichen eingelesen und auf 'hexadezimale Gültigkeit' überprüft. In diesem Fall wird der Akkumulator gelöscht und die unteren vier Bits werden abgetrennt. Diese vier Bits werden von rechts in EXPR geschoben. Die nachfolgende Zahl in EXPR wird auf die gleiche Art nach links geschoben.

Bei Eingabe einer Zahl mit mehr als vier Stellen werden nur die letzten vier Stellen berücksichtigt.

Beispiel: ABCDEF --> CDEF

```
                *        HEX-EINGABE PROGRAMM          *

                @$8A            =EXPR
                @$FFCF          =BASIN
                @$C000          =ORG
C000: A200      =HEXIN   LDX#   #0
C002: 868A               STX.   .EXPR
C004: 868B               STX.   .EXPR+1
C006: 20CFFF    =HEXIN1  JSR    ;BASIN
```

```
C009: C930            CMP#  '0
C00B: 901E            BCC   +HEXRTS
C00D: C93A            CMP#  ':
C00F: 900A            BCC   +HEXIN2
C011: C941            CMP#  'A
C013: 9016            BCC   +HEXRTS
C015: C947            CMP#  'G
C017: B012            BCS   +HEXRTS
C019: E936            SBC#  $36
C01B: 0A     =HEXIN2  ASLA
C01C: 0A              ASLA
C01D: 0A              ASLA
C01E: 0A              ASLA
C01F: A204            LDX#  #4
C021: 0A     =HEXIN3  ASLA
C022: 268A            ROL.  .EXPR
C024: 268B            ROL.  .EXPR+1
C026: CA              DEX
C027: D0F8            BNE   +HEXIN3
C029: F0DB            BEQ   +HEXIN1
C02B: 60     =HEXRTS  RTS

ASSEMBLIERUNG BEENDET
```

Hexadezimale Ausgabe

Das nächste Programm erklärt den Ausgabevorgang der berechneten Zahlen.

Sie wissen sicher noch, daß der Teil des Programms, der die Ausgabe bedient, ein Unterprogramm ist. Dieses Unterprogramm zeigt lediglich den Inhalt des Akkumulators auf dem Bildschirm. Das bedeutet, daß Sie zuerst den Akku mit dem Inhalt von EXPR+1 laden müssen und dann das Unterprogramm aufrufen, in dem zuerst die MSB- (most significant = höherwertigsten) und dann die LSB- (least significant = niederwertigsten) Bits ausgegeben werden.

```
                *        HEX-AUSGABE PROGRAMM          *

                @$8A            =EXPR
                @$FFD2          =BSOUT
                @$C000          =ORG
C000: A58B      =PRWORD  LDA.   .EXPR+1
C002: 200BB0             JSR    ;PRBYTE
C005: A58A               LDA.   .EXPR
C007: 200BB0             JSR    ;PRBYTE
C00A: 60                 RTS
C00B: 48        =PRBYTE  PHA
C00C: 4A                 LSRA
C00D: 4A                 LSRA
C00E: 4A                 LSRA
C00F: 4A                 LSRA
C010: 2016B0             JSR    ;HEXOUT
C013: 68                 PLA
C014: 290F               AND#   #15
C016: C90A      =HEXOUT  CMP#   #10
C018: B004               BCS    +ALFA
C01A: 0930               ORA#   '0
C01C: D002               BNE    +HXOUT
C01E: 6936      =ALFA    ADC#   #54
C020: 4CD2FF    =HXOUT   JMP    ;BSOUT

ASSEMBLIERUNG BEENDET
```

Dezimale Eingabe

Wahrscheinlich werden Sie beim Rechnen mit Zahlen die dezimalen Zahlen den hexadezimalen vorziehen. Das folgende Programm können Sie verwenden, um dezimale Zahlen für Rechner verständlich einzugeben.

Das Programm überprüft die Eingabe zuerst auf 'dezimale Richtigkeit' (0...9) bzw. bricht den Programmablauf ab, wenn ein anderes Zeichen eingegeben wurde. EXPR und EXPR+1 werden gelöscht. Als nächstes wird der Inhalt von EXPR und EXPR+1 mit 10 multipliziert und die neue Zahl addiert. Schließlich befinden sich die MSB-Bits

in EXPR+1 und die LSB-Bits in EXPR.

Zahlen, die größer sind als 65355 werden in Modulo 65536 (= der Rest, der beim (auch mehrmaligen) Abzug von 65536 übrigbleibt) angezeigt.

```
                *        DEZIMAL-EINGABE PROGRAMM      *

                @$8A            =EXPR
                @$FFCF          =BASIN
                @$C000          =ORG
C000: A200      =DECIN   LDX#   #0
C002: 868A               STX.   .EXPR
C004: 868B               STX.   .EXPR+1
C006: 20CFFF    =DEC1    JSR    ;BASIN
C009: C930               CMP#   '0
C00B: 9018               BCC    +DECEND
C00D: C93A               CMP#   ':
C00F: B014               BCS    +DECEND
C011: 290F               AND#   #15
C013: A211               LDX#   #17
C015: D005               BNE    +DEC3
C017: 9002      =DEC2    BCC    +DECDEC4
C019: 6909               ADC#   #9
C01B: 4A        =DEC4    LSRA
C01C: 668B      =DEC3    ROR.   .EXPR+1
C01E: 668A               ROR.   .EXPR
C020: CA                 DEX
C021: D0F4               BNE    +DEC2
C023: F0E1               BEQ    +DEC1
C025: 60        =DECEND  RTS
```

ASSEMBLIERUNG BEENDET

Dezimale Ausgabe

Das nächste Programm erlaubt die Anzeige von dezimalen Zahlen. Hier die Beschreibung:

Das X-Register wird mit dem ASCII-Äquivalent der Zahl 0 geladen. Diese Zahl wird zur höchsten Potenz von 10 (10000) erhoben und am Bildschirm angezeigt.

Der gleiche Vorgang wiederholt sich für 1000, 100 und 10. Der Rest wird mit Hilfe eines OR-Befehls in eine ASCII-Zahl umgewandelt und angezeigt.

Sie können diese Ausgaberoutine so abändern, daß die Anzeige führender Nullen unterdrückt wird.

```
                *        DEZIMAL-AUSGABE PROGRAMM      *

                @$8A            =DECL
                @$8B            =DECH
                @$8C            =TEMP
                @$FFD2          =BSOUT
                @$C000          =ORG
C000: A007      =DECOUT  LDY#    #7
C002: A230      =DECOUT1 LDX#   '0
C004: 38        =DECOUT2 SEC
C005: A58A               LDA.   .DECL
C007: F92EC0             SBCY   ;DECTAB-1
C00A: 48                 PHA
C00B: 88                 DEY
C00C: A58B               LDA.   .DECH
C00E: F930C0             SBCY   ;DECTAB+1
C011: 9009               BCC    +DECOUT3
C013: 858B               STA.   .DECH
C015: 68                 PLA
C016: 858A               STA.   .DECL
C018: E8                 INX
C019: C8                 INY
C01A: D0E8               BNE    +DECOUT2
C01C: 68        =DECOUT3 PLA
```

```
C01D: 8A                TXA
C01E: 848C              STY.   .TEMP
C020: 20D2FF            JSR    ;BSOUT
C023: A48C              LDY.   .TEMP
C025: 88                DEY
C026: 10DA              BPL    +DECOUT1
C028: A58A              LDA.   .DECL
C02A: 0930              ORA#   '0
C02C: 4CD2FF            JMP    ;BSOUT
C02F: 0A00    =DECTAB          ;#10
C031: 6400                     ;#100
C033: E803                     ;#1000
C035: 1027                     ;#10000

ASSEMBLIERUNG BEENDET
```

16 Bit Addition und Subtraktion

Die 16 Bit Addition ist bekannt. Sie wird hier noch einmal zusammen mit der Subtraktion aufgeführt.

```
              *        16 BIT ADDITION                *
              *        GANZZAHL OHNE VORZEICHEN       *
              *        EXPR1 := EXPR1 + EXPR2         *

              @$8A           =EXPR1
              @$8C           =EXPR2
              @$B000         =ORG
C000: 18      =ADD     CLC
C001: A58A             LDA.   .EXPR1
C003: 658C             ADC.   .EXPR2
C005: 858A             STA.   .EXPR1
C007: A58B             LDA.   .EXPR1+1
C009: 658D             ADC.   .EXPR2+1
C00B: 858B             STA.   .EXPR1+1
C00D: 60               RTS

ASSEMBLIERUNG BEENDET
```

```
                *        16 BIT SUBTRAKTION            *
                *        GANZZAHL OHNE VORZEICHEN      *
                *        EXPR1 := EXPR1 - EXPR2        *

                @$8A           =EXPR1
                @$8C           =EXPR2
                @$C000         =ORG
C000: 38        =SUB     SEC
C001: A58A               LDA.  .EXPR1
C003: E58C               SBC.  .EXPR2
C005: 858A               STA.  .EXPR1
C007: A58B               LDA.  .EXPR1+1
C009: E58D               SBC.  .EXPR2+1
C00B: 858B               STA.  .EXPR1+1
C00D: 60                 RTS

ASSEMBLIERUNG BEENDET
```

16 Bit Multiplikation

Die Multiplikation ist wesentlich komplizierter als die Addition oder Subtraktion. Sie ist im binären Zahlensystem mit der im dezimalen Zahlensystem identisch. Sehen wir doch einmal, wie wir im dezimalen System multiplizieren. Als Beispiel multiplizieren wir 5678*203.

```
        5678
         203 *
        ------
       17034
      00000
     11356
   -----------
     1252634
```

Mit jeder Stelle wird die nachfolgende Zahl nach rechts geschoben. Ist die Stelle ungleich Null, wird das neue Zwischenergebnis addiert. Im binären System funktioniert das auf die gleiche Weise.

Ein- und Ausgabe von Zahlen

Ein Beispiel:

```
      1011
      1101 *
      ------
      1011
     0000
    1011
   1011
  ----------
  10001111
```

Wie Sie sehen, ist die Multiplikation im binären System einfacher als im dezimalen System. Da die größtmögliche Zahl einer jeden Stelle nur 1 sein kann, ist das höchste Zwischenergebnis gleich dem Multiplikant.

Das folgende Programm läuft im Prinzip genauso ab wie eben beschrieben, nur wird hier das Zwischenergebnis nach rechts geschoben und der Multiplikant wird, wenn nötig, addiert.

Es werden sechs Speicherplätze benötigt. Zwei davon (SCRATCH und SCRATCH+1) werden nur zeitweise benutzt, die restlichen vier enthalten die zwei Zahlen, die multipliziert werden sollen (EXPR1 und EXPR1+1, EXPR2 und EXPR2+1). Nach der Berechnung steht das Ergebnis in den Speicherstellen EXPR1(LSB) und EXPR1+1(MSB).

```
                *        16 BIT MULTPLIKATION          *
                *        GANZZAHL OHNE VORZEICHEN      *
                *        EXPR1 := EXPR1 * EXPR2        *

                @$8A           =EXPR1
                @$8C           =EXPR2
                @$8E           =SCRATCH
                @$C000         =ORG
C000: A200      =MUL     LDX#  #0
C002: 868E               STX.  .SCRATCH
C004: 868F               STX.  .SCRATCH+1
C006: A010               LDY#  #16
C008: D00D               BNE   +MUL2
C00A: 18        =MUL1    CLC
C00B: A58E               LDA.  .SCRATCH
```

```
C00D: 658C            ADC.  .EXPR2
C00F: 858E            STA.  .SCRATCH
C011: A58F            LDA.  .SCRATCH+1
C013: 658D            ADC.  .EXPR2+1
C015: 858F            STA.  .SCRATCH+1
C017: 468F    =MUL2   LSR.  .SCRATCH+1
C019: 668E            ROR.  .SCRATCH
C01B: 668B            ROR.  .EXPR1+1
C01D: 668A            ROR.  .EXPR1
C01F: 88              DEY
C020: 3004            BMI   +MULRTS
C022: 90F3            BCC   +MUL2
C024: B0E4            BCS   +MUL1
C026: 60      =MULRTS RTS

ASSEMBLIERUNG BEENDET
```

16 Bit Division

Die Division zweier Zahlen ist gerade das Gegenteil der Multiplikation. Sie können deshalb auch im nachfolgenden Programm sehen, daß der Divisor subtrahiert wird und der Dividend nach links geschoben wird. Die verwendeten Speicherstellen sind dieselben wie beim Multiplikationsprogramm, nur werden die Stellen SCRATCH und SCRATCH+1 jetzt REMAIN und REMAIN+1 genannt. Das bedeutet, daß der Rest der Division in diese Speicherstellen abgelegt wird.

```
              *        16 BIT DIVISION                *
              *        GANZZAHL OHNE VORZEICHEN       *
              *        EXPR1 := EXPR1 DURCH EXPR2     *
              *        REMAIN := EXPR1 MOD EXPR2      *

              @$8A           =EXPR1
              @$8C           =EXPR2
              @$8E           =REMAIN
              @$C000         =ORG
C000: A200    =DIV     LDX#  #0
```

```
C002: 868E            STX.  .REMAIN
C004: 868F            STX.  .REMAIN+1
C006: AC3C01          LDY   ;316
C009: 068A   =DIV1    ASL.  .EXPR1
C00B: 268B            ROL.  .EXPR1+1
C00D: 268E            ROL.  .REMAIN
C00F: 268F            ROL.  .REMAIN+1
C011: 38              SEC
C012: A58E            LDA.  .REMAIN
C014: E58C            SBC.  .EXPR2
C016: AA              TAX
C017: A58F            LDA.  .REMAIN+1
C019: E58D            SBC.  .EXPR2+1
C01B: 9006            BCC    +DIV2
C01D: 868E            STX.  .REMAIN
C01F: 858F            STA.  .REMAIN+1
C021: E68A            INC.  .EXPR1
C023: 88     =DIV2    DEY
C024: D0E3            BNE   +DIV1
C026: 60              RTS
```

ASSEMBLIERUNG BEENDET

2

Textausgabe

Textausgabe

Bei den meisten Programmen muß Text ausgegeben werden (Menüs usw.).

Das nachfolgende Programm ermöglicht die Ausgabe von Zeichenketten beliebiger Länge an beliebigen Positionen. Der Ausgabebefehl kann an jeder Stelle Ihres Programms stehen.

Wie funktioniert das?

Der Mikroprozessor 6502 benutzt seinen Stack dazu, die Rückkehradresse abzuspeichern, wenn ein JSR-Befehl ausgeführt wird. Die auf dem Stack abgelegte Zahl entspricht der Rückkehradresse minus Eins. Der in diesem Programm verwendete Trick besteht darin, daß die auszudruckende Zeichenkette unmittelbar nach dem JSR-Befehl folgt und das letzte Zeichen der Kette um 128 erhöht wird. Das Unterprogramm berechnet die Anfangsadresse der Zeichenkette mit Hilfe der Zahl auf dem Stack und liest die Zeichen Byte für Byte, bis es das um 128 erhöhte Byte gefunden hat. Die Adresse dieses Bytes wird auf dem Stack abgelegt und ein RTS-Befehl ausgeführt. So wird die Zeichenkette übersprungen und das nächste Kommando ausgeführt.

```
                *        AUSGABE VON ZEICHENKETTEN     *
                *        MIT UNTERSCHIEDLICHER LAENGE  *

                @2               =AUX
                @$FFD2           =BSOUT
                @$C000           =ORG
C000: 2014B0 =EXAMPLE JSR        ;PRINT
C003:                            /DAS IST EIN TEST
C003: 444153
C006: 204953
C009: 542045
C00C: 494E20
C00F: 544553
C012: D4
C013: 00                 BRK
C014: 68        =PRINT   PLA
C015: 8502               STA.    .AUX
```

```
C017: 68                PLA
C018: 8503              STA.   .AUX+1
C01A: A200              LDX#   #0
C01C: E602    =PRINT1   INC.   .AUX
C01E: D002              BNE    +PRINT2
C020: E603              INC.   .AUX+1
C022: A102    =PRINT2   LDA@X  .AUX
C024: 297F              AND#   $7F
C026: 20D2FF            JSR    ;BSOUT
C029: A200              LDX#   #0
C02B: A102              LDA@X  .AUX
C02D: 10ED              BPL    +PRINT1
C02F: A503              LDA.   .AUX+1
C031: 48                PHA
C032: A502              LDA.   .AUX
C034: 48                PHA
C035: 60                RTS

ASSEMBLIERUNG BEENDET
```

3

Rekursion

Rekursion

Eine Programmiermethode, die immer häufiger angewendet wird, ist die sogenannte rekursive Programmierung. Was ist Rekursion?

Rekursion wird häufig in der Mathematik benutzt. Als Beispiel sei die Fakultätsfunktion angeführt, die mit Rekursion folgendermaßen definiert werden kann:

```
0! = 1  ;

n! = n * (n-1)!
```

Mit diesen zwei Definitionen ist die gesamte Funktion beschrieben und kann ausgeführt werden. Als Beispiel wollen wir 6! berechnen. Wir verwenden dazu die nachfolgende Rekursion:

```
     6! = 6*5!
     6! = 6*(5*4!)
     6! = 6*(5*(4*3!))
     6! = 6*(5*(4*(3*2!)))
     6! = 6*(5*(4*(3*(2*1!))))
     6! = 6*(5*(4*(3*(2*(1*0!)))))
                          :
                          +- 0! = 1

-->  6! = 6*5*4*3*2*1*1 = 720
```

Wir haben also die Funktion durch das schrittweise Ausführen unserer Definition berechnet. Die Berechnung wurde durch das Erreichen der Definition von 0! abgebrochen. Diese Definition ist sehr wichtig, sonst würden wir immer weiterrechnen.

Warum brauchen wir Rekursionen beim Programmieren?

Wir haben gesehen, daß Funktionen mit Hilfe der rekursiven Programmierung leicht zu berechnen sind.

Viele Probleme kann man mit Rekursionen und einfachen Algorithmen lösen. Allerdings muß man sich erst daran gewöhnen, daß das

Programm sich selbst aufruft. Eine allgemeine Anweisung in einer Hochsprache könnte folgendermaßen aussehen:

```
PROC R:
IF C
THEN D
ELSE S;  R
FI
```

Ist Bedingung C wahr, wird das Programm die Anweisung D ausführen und anhalten. Sonst wird S ausgeführt und das Programm ruft sich selbst (R) wieder auf. Unser Beispiel könnte dann so aussehen:

```
PROC fac IN n OUT f
IF n = 0
THEN f= 1
ELSE f = n * fac( n-1 )
FI
```

Diese Prozedur (Unterprogramm) ruft sich solange selbst auf, bis n=0 ist und bringt 1 als Ergebnis. Sie kehrt dann zum vorhergehenden Aufruf zurück usw. Schließlich wird der erste Aufruf von 'fac' als Ergebnis n*(n-1)*(n-2)*...*(1)*1 bringen, also den Wert von n!.

Die nächsten zwei Programme zeigen eine weitere Anwendung von Rekursion. Die Ausgabe einer binären Zahl in hexadezimaler Form. Das erste Programm druckt den Inhalt der Speicherstellen EXPR (LSB) und EXPR+1 (MSB) als vorzeichenlose Zahl aus. Das Programm wird beendet, wenn der Inhalt von EXPR und EXPR+1 kleiner als 10 ist. Sonst wird durch 10 dividiert und das Programm ruft sich selbst wieder auf. Zuerst wird die höchstwertigste Stelle ausgegeben, dann die nachfolgenden. Dieser Vorgang beseitigt automatisch etwaige führende Nullen.

```
*          REKURSION PUTINT            *
*          GANZZAHL OHNE VORZEICHEN    *

@2              =EXPR
@4              =REMAIN
@$FFD2          =BSOUT
@13             =CR
@$C000          =ORG
```

```
                * 65535
C000: A9FF              LDA#  #255
C002: 8502              STA.  .EXPR
C004: A9FF              LDA#  #255
C006: 8503              STA.  .EXPR+1
C008: 3051C0            JSR   ;PUTINT
C00B: A90D              LDA#  .CR
C00D: 20D2FF            JSR   ;BSOUT
                * 0
C010: A900              LDA#  #0
C012: 8502              STA.  .EXPR
C014: A900              LDA#  #0
C016: 8503              STA.  .EXPR+1
C018: 3051C0            JSR   ;PUTINT
C01B: A90D              LDA#  .CR
C01D: 20D2FF            JSR   ;BSOUT
                * -32768
C020: A900              LDA#  #0
C022: 8502              STA.  .EXPR
C024: A980              LDA#  #128
C026: 8503              STA.  .EXPR+1
C028: 3051C0            JSR   ;PUTINT
C02B: A90D              LDA#  .CR
C02D: 20D2FF            JSR   ;BSOUT
                * 12345
C030: A939              LDA#  #57
C032: 8502              STA.  .EXPR
C034: A930              LDA#  #48
C036: 8503              STA.  .EXPR+1
C038: 3051C0            JSR   ;PUTINT
C03B: A90D              LDA#  .CR
C03D: 20D2FF            JSR   ;BSOUT
                * -1
C040: A9FF              LDA#  #255
C042: 8502              STA.  .EXPR
C044: A9FF              LDA#  #255
C046: 8503              STA.  .EXPR+1
C048: 3051C0            JSR   ;PUTINT
C04B: A90D              LDA#  .CR
C04D: 20D2FF            JSR   ;BSOUT
C050: 00                BRK
C051: 48        =PUTINT PHA
C052: A503              LDA.  .EXPR+1
C054: D006              BNE   +NODIG
```

```
C056: A502            LDA.  .EXPR
C058: C90A            CMP#  #10
C05A: 9006            BCC   +PUTDIG
C05C: 2069C0 =NODIG   JSR   ;DIV10
             * SPRINGE IN REKURSION
C05F: 2051C0          JSR   ;PUTINT
C062: 0930   =PUTDIG  ORA#  '0
C064: 20D2FF          JSR   ;BSOUT
C067: 68              PLA
C068: 60              RTS
             * DIVISION DURCH ZEHN, PROGRAMM AUS KAPITEL 1
C069: A200   =DIV10   LDX#  #0
C06B: 868E            STX.  .REMAIN
C06D: 868F            STX.  .REMAIN+1
C06F: A010            LDY#  #16
C071: 0602   DIV1     ASL.  .EXPR
C073: 2603            ROL.  .EXPR+1
C075: 2604            ROL.  .REMAIN
C077: 2605            ROL.  .REMAIN+1
C079: 38              SEC
C07A: A504            LDA.  .REMAIN
C07C: E90A            SBC#  #10
C07E: AA              TAX
C07F: A505            LDA.  .REMAIN+1
C081: E900            SBC#  #0
C083: 9006            BCC   +DIV2
C085: 8604            STX.  .REMAIN
C087: 8505            STA.  .REMAIN+1
C089: E602            INC.  .EXPR
C08B: 88     =DIV2    DEY
C08C: D0E3            BNE   +DIV1
C08E: A504            LDA.  .REMAIN
C090: 60              RTS

ASSEMBLIERUNG BEENDET
```

Mit ein paar Instruktionen kann man das Programm so umwandeln, daß es Ganzzahlen mit Vorzeichen ausdruckt. Zuerst wird das Vorzeichen der zu druckenden Zahl geprüft und ein Leerzeichen (>=0) oder ein Minuszeichen (<0) ausgedruckt. Ist die Zahl kleiner Null, wird sie invertiert und wie eine vorzeichenlose Zahl ausgegeben. Damit können Sie Zahlen zwischen -32768 und 32767

ausdrucken.

```
                *          REKURSION PUTINT              *
                *          GANZZAHL MIT VORZEICHEN       *

                @2                =EXPR
                @4                =REMAIN
                @$FFD2            =BSOUT
                @13               =CR
                @$C000            =ORG
                * 65535
C000: A9FF              LDA#  #255
C002: 8502              STA.  .EXPR
C004: A9FF              LDA#  #255
C006: 8503              STA.  .EXPR+1
C008: 3051C0            JSR   ;PUTINT
C00B: A90D              LDA#  .CR
C00D: 20D2FF            JSR   ;BSOUT
                * 0
C010: A900              LDA#  #0
C012: 8502              STA.  .EXPR
C014: A900              LDA#  #0
C016: 8503              STA.  .EXPR+1
C018: 3051C0            JSR   ;PUTINT
C01B: A90D              LDA#  .CR
C01D: 20D2FF            JSR   ;BSOUT
                * -32768
C020: A900              LDA#  #0
C022: 8502              STA.  .EXPR
C024: A980              LDA#  #128
C026: 8503              STA.  .EXPR+1
C028: 3051C0            JSR   ;PUTINT
C02B: A90D              LDA#  .CR
C02D: 20D2FF            JSR   ;BSOUT
                * 32767
C030: A9FF              LDA#  #255
C032: 8502              STA.  .EXPR
C034: A97F              LDA#  #127
C036: 8503              STA.  .EXPR+1
C038: 3051C0            JSR   ;PUTINT
C03B: A90D              LDA#  .CR
```

```
C03D: 20D2FF          JSR   ;BSOUT
            * 12345
C040: A939            LDA#  #57
C042: 8502            STA.  .EXPR
C044: A930            LDA#  #48
C046: 8503            STA.  .EXPR+1
C048: 3051C0          JSR   ;PUTINT
C04B: A90D            LDA#  .CR
C04D: 20D2FF          JSR   ;BSOUT
            * -1
C050: A9FF            LDA#  #255
C052: 8502            STA.  .EXPR
C054: A9FF            LDA#  #255
C056: 8503            STA.  .EXPR+1
C058: 3051C0          JSR   ;PUTINT
C05B: A90D            LDA#  .CR
C05D: 20D2FF          JSR   ;BSOUT
            * -345
C060: A9A7            LDA#  #167
C062: 8502            STA.  .EXPR
C064: A9FE            LDA#  #254
C066: 8503            STA.  .EXPR+1
C068: 3051C0          JSR   ;PUTINT
C06B: A90D            LDA#  .CR
C06D: 20D2FF          JSR   ;BSOUT
C070: 00              BRK
C071: A920   =PUTINTS LDA#  '
C073: 2403            BIT.  .EXPR+1
C075: 100F            BPL   +PUTSGN
            * WENN<0 --> 2'ER KOMPLEMENT UND DRUCKE '-' AUS
C077: 38              SEC
C078: A900            LDA#  #0
C07A: E502            SBC.  .EXPR
C07C: 8502            STA.  .EXPR
C07E: A900            LDA#  #0
C080: E503            SBC.  .EXPR+1
C082: 8503            STA.  .EXPR+1
C084: A92D            LDA#  '-
C086: 20D2FF =PUTSGN  JSR   ;BSOUT
C089: 48     =PUTINT  PHA
C08A: A503            LDA.  .EXPR+1
C08C: D006            BNE   +NODIG
C08E: A502            LDA.  .EXPR
C090: C90A            CMP#  #10
```

```
C092: 9006              BCC   +PUTDIG
C094: 20A1C0 NODIG      JSR   ;DIV10
             * SPRINGE IN REKURSION
C097: 2089C0            JSR   ;PUTINT
C09A: 0930   =PUTDIG    ORA#  #'0
C09C: 20D2FF            JSR   ;BSOUT
C09F: 68                PLA
C0A0: 60                RTS
             * DIVISION DURCH ZEHN PROGRAMM AUS KAPITEL 1
C0A1: A200   =DIV10     LDX#  #0
C0A3: 868E              STX.  .REMAIN
C0A5: 868F              STX.  .REMAIN+1
C0A7: A010              LDY#  #16
C0A9: 0602   =DIV1      ASL.  .EXPR
C0AB: 2603              ROL.  .EXPR+1
C0AD: 2604              ROL.  .REMAIN
C0AF: 2605              ROL.  .REMAIN+1
C0B1: 38                SEC
C0B2: A504              LDA.  .REMAIN
C0B4: E90A              SBC#  #10
C0B6: AA                TAX
C0B7: A505              LDA.  .REMAIN+1
C0B9: E900              SBC#  #0
C0BB: 9006              BCC   +DIV2
C0BD: 8604              STX.  .REMAIN
C0BF: 8505              STA.  .REMAIN+1
C0C1: E602              INC.  .EXPR
C0C3: 88     =DIV2      DEY
C0C4: D0E3              BNE   +DIV1
C0C6: A504              LDA.  .REMAIN
C0C8: 60                RTS

ASSEMBLIERUNG BEENDET
```

4

Das Dateisystem

Das Dateisystem

Der Commodore 64 verwendet das bei Commodore Rechnern übliche Dateisystem. Dieses System kann gleichzeitig bis zu 10 Dateien verwalten. In BASIC gibt es genügend Unterstützung für die Benutzung der Ein- und Ausgabe (=IO: OPEN, PRINT# USW.). Wie kann man das in Maschinensprache erreichen?

Das OS (Operating System = Betriebssystem) übergibt Ihnen die höchste ROM-Speicherstelle in einer Sprungleiste (über Vektoren), damit Sie die verschiedenen Unterprogramme leicht benutzen können.

Nachfolgend eine kurze Erklärung der verschiedenen Routinen.

Name : SETFPA
Adresse : $FFBA
beeinflußt : nichts

Setzt die Dateiparameter. Die logische Dateinummer muß im Akkumulator sein. Das X-Register enthält die Gerätenummer und das Y-Register die Sekundäradresse.

Name : SETFNA
Adresse : $FFBD
beeinflußt : nichts

Setzt den Dateinamen. Der Akku enthält die Länge des Namens, das X-Register das niederwertige Byte der Adresse des Dateinamens und das Y-Register das höherwertige Byte.

Name : OPEN
Adresse : $FFC0
beeinflußt : alle Register

'Eröffnet' eine Datei. Zuerst müssen die Dateiparameter und der Dateiname übergeben sein. Ist das Carry-Flag gesetzt, dann wurde während des 'Eröffnens' der Datei ein Fehler festgestellt.

Name : CLOSE
Adresse : $FFC3
beeinflußt : alle Register

'Schließt' eine Datei. Der Akku muß die Dateinummer der zu schließenden Datei enthalten.

Name : CHKIN
Adresse : $FFC6
beeinflußt : alle Register

Ermöglicht die Eingabe in eine logische Datei/Gerät. Das X-Register beinhaltet die Dateinummer einer schon 'geöffneten' Datei. Das Betriebssystem (OS) wird von nun an Daten von dem Gerät, das der Datei zugeordnet ist, einlesen. Ein gesetztes Carry-Flag zeigt einen Fehler an.

Name : CKOUT
Adresse : $FFC9
beeinflußt : alle Register

Ermöglicht die Ausgabe zu einer Datei bzw. einem Gerät. Das X-Register enthält die Dateinummer einer schon 'eröffneten' Datei. Das Betriebssystem (OS) wird von nun an Daten zum Gerät, das der Datei zugeordnet ist, ausgeben. Ein gesetztes Carry-Flag zeigt einen Fehler an.

Name : CLRCH
Adresse : $FFCC
beeinflußt : alle Register

Setzt die Ein- und Ausgabedateien auf die Standarddateien zurück. Eingabe von der Tastatur (Gerät 0) und Ausgabe auf den Bildschirm (Gerät 3).

Name : BASIN
Adresse : $FFCF
beeinflußt : Y und Akku

Byteeingabe vom Eingabegerät; erfolgt standardmäßig von der Ta-

statur. BASIN wartet auf einen Wagenrücklauf (CR = carriage return) und gibt dann alle Bytes über den Akku einschließlich des Wagenrücklaufs (CR) aus. Gesetztes Carry zeigt einen Fehler an.

Name : BSOUT
Adresse : $FFD2
beeinflußt : nichts

Byteausgabe zum Ausgabegerät; erfolgt standardmäßig auf den Bildschirm. Der Akku muß das Byte enthalten, das ausgegeben werden soll. Gesetztes Carry zeigt einen Fehler an.

Name : GET
Adresse : $FFE4
beeinflußt : Y und Akku

Holt ein Byte vom Eingabegerät. Erfolgt standardmäßig von der Tastatur. Sind keine Daten vorhanden, wird eine Null zurückgegeben. Gesetztes Carry zeigt einen Fehler an.

Name : CLALL
Adresse : $FFE7
beeinflußt : alle Register

'Schließt' alle Dateien und führt einen CLRCH durch.

Die folgenden Rumpfprogramme zeigen Ihnen, wie Sie auf verschiedenen Geräten in Maschinensprache Dateien 'eröffnen' können. Die equivalenten BASIC-Anweisungen werden mitangegeben.

```
*        OEFFNEN DER LOGISCHEN        *
*        DATEI KASSETTE ZUR EINGABE   *
*        IN BASIC: OPEN1,1,0,"NAME"   *

@$FFBA        =SETFPA
@$FFBD        =SETFNA
@FFC0         =OPEN
@4            =FNLEN
```

```
            @$C000          =ORG
C000: 4C07C0          JMP   ;FNAME+4
            * DER DATEINAME
C003:       =FNAME          &NAME
C003: 4E414D
C006: 45
C007: A901            LDA#  #1
C009: A201            LDX#  #1
C00B: A000            LDY#  #0
C00D: 20BAFF          JSR   ;SETFPA
C010: A203            LDX#  $3
C012: A0C0            LDY#  $C0
C014: A904            LDA#  .FNLEN
C016: 20BDFF          JSR   ;SETFNA
C019: 20C0FF          JSR   ;OPEN
            * ANWENDER-PROGRAMM
            *
            *
```

ASSEMBLIERUNG BEENDET

```
            *       OEFFNEN DER LOGISCHEN          *
            *       DATEI KASSETTE ZUR AUSGABE     *
            *       IN BASIC: OPEN1,1,1,"NAME"     *

            @$FFBA          =SETFPA
            @$FFBD          =SETFNA
            @$FFC0          =OPEN
            @4              =FNLEN
            @$C000          =ORG
C000: 4C07C0          JMP   ;FNAME+4
            * DER DATEINAME
C003:       =FNAME          &NAME
C003: 4E414D
C006: 45
C007: A901            LDA#  #1
C009: A201            LDX#  #1
C00B: A001            LDY#  #1
C00D: 20BAFF          JSR   ;SETFPA
C010: A203            LDX#  $3
C012: A0C0            LDY#  $C0
C014: A904            LDA#  #FNLEN
```

```
C016: 20BDFF          JSR   ;SETFNA
C019: 20C0FF          JSR   ;OPEN
            * ANWENDER-PROGRAMM
            *

ASSEMBLIERUNG BEENDET


            *         OEFFNEN DER LOGISCHEN         *
            *         DATEI DISKETTE ZUR EINGABE    *
            *         IN BASIC: OPEN1,8,6,"0:NAME"  *

            @$FFBA          =SETFPA
            @$FFBD          =SETFNA
            @$FFC0          =OPEN
            @6              =FNLEN
            @$C000          =ORG
C000: 4C09C0          JMP   ;FNAME+6
            * DER DATEINAME
C003:       =FNAME          &0:NAME
C003: 303A4E
C006: 414D45
C009: A901            LDA#  #1
C00B: A208            LDX#  #8
C00D: A006            LDY#  #6
C00F: 20BAFF          JSR   ;SETFPA
C012: A203            LDX#  $3
C014: A0C0            LDY#  $C0
C016: A906            LDA#  #FNLEN
C018: 20BDFF          JSR   ;SETFNA
C01B: 20C0FF          JSR   ;OPEN
            * ANWENDER-PROGRAMM
            *

ASSEMBLIERUNG BEENDET
```

```
                *        OEFFNEN DER LOGISCHEN          *
                *        DATEI DISKETTE ZUR AUSGABE     *
                *        IN BASIC: OPEN1,8,6,"0:NAME,S,W"

                @$FFBA          =SETFPA
                @$FFBD          =SETFNA
                @$FFC0          =OPEN
                @10             =FNLEN
                @$C000          =ORG
C000: 4C0DC0            JMP    ;FNAME+10
                * DER DATEINAME
C003:           =FNAME          &0:NAME,S,W
C003: 303A4E
C006: 414D45
C009: 2C532C
C00C: 57
C00D: A901              LDA#   #1
C00F: A208              LDX#   #8
C011: A006              LDY#   #6
C013: 20BAFF            JSR    ;SETFPA
C016: A203              LDX#   $3
C018: A0C0              LDY#   $C0
C01A: A904              LDA#   #FNLEN
C01C: 20BDFF            JSR    ;SETFNA
C020: 20C0FF            JSR    ;OPEN
                * ANWENDER-PROGRAMM
                *

ASSEMBLIERUNG BEENDET
```

Oft soll eine Ein- oder Ausgabe mit der STOP-Taste unterbrochen werden können. Das OS unterstützt das mit einem Unterprogramm, genannt STOPQ ($FFE1). Das Zero-Flag wird gesetzt, wenn die STOP-Taste gedrückt wird. Mit BEQ und BNE können Sie nun leicht entscheiden, welchen Weg Sie in Ihrem Programm weitergehen wollen.

Abspeichern und Laden von Programmen im OS

Wünschen Sie einen Bereich des Speichers abzuspeichern oder zu laden, können Sie die schon vorhandenen Unterprogramme SAVE und LOAD benutzen.

Name : LOAD
Adresse : $FFD5
beeinflußt : alle Register

Lädt ein Programm in den Speicher (X enthält das niederwertige, Y das höherwertige Byte der Anfangsadresse). Der Akku zeigt an, ob ein 'verify' (A=1) oder ein 'load' (A=0) durchgeführt werden soll. Notwendigerweise müssen zuerst wieder die verschiedenen Parametertypen (Gerät, Sekundäradresse) und der Dateiname gesetzt werden. Ist die Sekundäradresse gleich Null, wird das Programm wie ein relatives BASIC-Programm geladen. Bei einer 1 als Sekundäradresse wird absolut geladen. Gesetztes Carry zeigt einen Fehler an.

Name : SAVE
Adresse : $FFD8
beeinflußt : alle Register

Speichert ein Programm ab (X enthält das niederwertige Byte der Endadresse, Y das höherwertige und der Akku zeigt auf die zwei Speicherstellen in der Zeropage, die die Startadresse enthalten). Auch hier müssen die Dateiparameter und der Dateiname bereits gesetzt sein. Sie können das Programm mit unterschiedlichen Sekundäradressen abspeichern:

sa = 0 abgespeichert als BASIC-Programm
sa = 1 abgespeichert als Maschinenspracheprogramm
sa = 2 abgespeichert als BASIC-Programm mit EOTBlock
sa = 3 abgespeichert als Maschinenspracheprogramm mit EOT

Das gesetzte Carrybit zeigt einen Fehler an.

Die folgenden Programmbeispiele gelten für Diskette und Kassette.

```
                *        EIN PROGRAMM VON                *
                *        DER KASSETTE LADEN              *

                @2              =START
                @$FFBA          =SETFPA
                @$FFBD          =SETFNA
                @$FFD2          =BSOUT
                @$FFD5          =LOAD
                @0              =SECADR
                @0              =LOADF
                @1              =DEVNUM
                @$C100          =FNAME
C100:                           &TEST
C100: 544553
C103: 54
                @4              =FLEN
                @$C000          =ORG
                * SEKUNDAERADRESSE UND GERAETENUMMER SETZEN
C000: A000              LDY#  .SECADR
C002: A201              LDX#  .DEVNUM
C004: 20BAFF            JSR   ;SETFPA
                * DATEINAME UND LAENGE
C006: A200              LDX#  $0
C008: A0C1              LDY#  $C1
C00A: A904              LDA#  #FLEN
C00D: 20BDFF            JSR   ;SETFNA
                * ADRESSE, AB DER DAS PROGRAMM STEHEN SOLL
                * AKKU=0 FUER LOAD
C010: A602              LDX.  .START
C012: A403              LDY.  .START+1
C014: A900              LDA#  .LOADF
C016: 20D5FF            JSR   ;LOAD
C019: B001              BCS   +LOADERR
C01B: 00                BRK
                * C=1 --> FEHLER AUSGABE VON '?'
C01C: A93F      =LOADERR LDA#  #'?
C01E: 20D2FF            JSR   ;BSOUT
C021: 00                BRK

ASSEMBLIERUNG BEENDET
```

```
                *        EIN PROGRAMM VON                *
                *        DER DISKETTE LADEN              *

                @2              =START
                @$FFBA          =SETFPA
                @$FFBD          =SETFNA
                @$FFD2          =BSOUT
                @$FFD5          =LOAD
                @0              =SECADR
                @0              =LOADF
                @8              =DEVNUM
                @$C100          =FNAME
C100:                           &0:TEST
C100: 303A54
C103: 455354
                @6              =FLEN
                @$C000          =ORG
                * SEKUNDAERADRESSE UND GERAETENUMMER SETZEN
C000: A000              LDY#  .SECADR
C002: A208              LDX#  .DEVNUM
C004: 20BAFF            JSR   ;SETFPA
C006: A200              LDX#  $0
C008: A0C1              LDY#  $C1
C00A: A906              LDA#  .FLEN
C00D: 20BDFF            JSR   ;SETFNA
                * ADRESSE, AB DER DAS PROGRAMM STEHEN SOLL
                * AKKU=0 FUER LOAD
C010: A602              LDX.  .START
C012: A403              LDY.  .START+1
C014: A900              LDA#  .LOADF
C016: 20D5FF            JSR   ;LOAD
C019: B001              BCS   +LOADERR
C01B: 00                BRK
                * C=1 --> FEHLER AUSGABE VON '?'
C01C: A93F    =LOADERR LDA#  '?
C01E: 20D2FF            JSR   ;BSOUT
C021: 00                BRK

ASSEMBLIERUNG BEENDET
```

```
            *        EIN PROGRAMM AUF              *
            *        DIE KASSETTE SPEICHERN        *

            @2             =START
            @4             =END
            @$FFBA         =SETFPA
            @$FFBD         =SETFNA
            @$FFD2         =BSOUT
            @$FFD8         =SAVE
            @1             =SECADR
            @1             =DEVNUM
            @$C100         =FNAME
C100:                      &TEST
C100: 544553
C103: 54
            @4             =FLEN
            @$C000         =ORG
C000: A001            LDY#  .SECADR
C002: A201            LDX#  .DEVNUM
C004: 20BAFF          JSR   ;SETFPA
C006: A200            LDX#  $0
C008: A0C1            LDY#  $C1
C00A: A904            LDA#  .FLEN
C00D: 20BDFF          JSR   ;SETFNA
            * ENDADRESSE IN (X,Y), ZEIGER AUF DIE START-
            * ADRESSE IM AKKU
C010: A604            LDX.  .END
C012: A405            LDY.  .END+1
C014: A902            LDA#  .START
C016: 20D8FF          JSR   ;SAVE
C019: B001            BCS   +SAVEERR
C01B: 00              BRK
            * C=1 --> FEHLER AUSGABE VON '?'
C01C: A93F   =SAVEERR LDA#  '?
C01E: 20D2FF          JSR   ;BSOUT
C021: 00              BRK

ASSEMBLIERUNG BEENDET
```

```
                *        EIN PROGRAMM AUF                *
                *        DIE DISKETTE SPEICHERN          *

               @2                =START
               @4                =END
               @$FFBA            =SETFPA
               @$FFBD            =SETFNA
               @$FFD2            =BSOUT
               @$FFD8            =SAVE
               @1                =SECADR
               @8                =DEVNUM
               @$C100            =FNAME
C100:                            &0:TEST
C100: 303A54
C103: 455354
               @6                =FLEN
               @$C000            =ORG
C000: A001              LDY#  .SECADR
C002: A208              LDX#  .DEVNUM
C004: 20BAFF            JSR   ;SETFPA
C006: A200              LDX#  $0
C008: A0C1              LDY#  $C1
C00A: A906              LDA#  .FLEN
C00D: 20BDFF            JSR   ;SETFNA
               * ENDADRESSE IN (X,Y), ZEIGER AUF DIE START-
               * ADRESSE IM AKKU
C010: A604              LDX.  .END
C012: A405              LDY.  .END+1
C014: A902              LDA#  .START
C016: 20D8FF            JSR   ;SAVE
C019: B001              BCS   +SAVEERR
C01B: 00                BRK
               * C=1 --> FEHLER, AUSGABE VON '?'
C01C: A93F     =SAVEERR LDA#  '?
C01E: 20D2FF            JSR   ;BSOUT
C021: 00                BRK

ASSEMBLIERUNG BEENDET
```

5

Die RS232-Schnittstelle

Die RS232-Schnittstelle

Der Commodore 64 besitzt eine eingebaute RS232-Schnittstele, welche es Ihnen ermöglicht, Printer, Modems und Terminals anzuschließen. Die RS232-Schnittstelle verwendet das User Port an der Rückseite des C64. Verwechseln Sie es bitte nicht mit dem Erweiterungsport an der rechten Seite. Die Pinbelegung am User Port sieht folgendermaßen aus:

PIN Besch	6526 Beschr.	Beschreibung	ein/ aus
C	PB0	Empfangsdaten	ein
D	PB1	request to send	aus
E	PB2	data terminal ready	aus
F	PB3	ring indicator	ein
H	PB4	received line signal	ein
J	PB5	nicht belegt	ein
K	PB6	data set ready	ein
B	FLG2	Empfangsdaten	ein
M	PA2	Sendedaten	aus
A	GND	Schutzerde	
N	GND	Signalerde	

Wie Sie sicher bemerkt haben, gibt es zwei Leitungen, die die gleiche Bezeichnung haben: Empfangsdaten. Damit die RS232-Schnittstelle auch richtig funktioniert, müssen beide verbunden

werden. Außerdem müssen die Signale auf den für die RS232-Schnittstelle vorgeschriebenen Spannungspegel gebracht und einige Signale invertiert werden. Eine vollständige Anpassung der Signale an die RS232-Bedingungen erhalten Sie durch das von Commodore angebotene Steckmodul.

Die RS232-Schnittstelle benötigt einen Dateinamen, der aus zwei Byte besteht. Die Anordnung der Bits in diesen Bytes beeinflußt ganz wesentlich die Art, wie die Schnittstelle arbeitet. In BASIC müssen Sie die Werte dieser Bytes in zwei dezimale Zahlen umwandeln und diese Zahlen mit Hilfe des CHR$-Befehls als Dateiname eingeben. In Maschinensprache müssen X- und Y-Register die Adresse enthalten, an der die beiden Bytes (der Dateiname) stehen. Der Akku muß mit 2 (Länge des Dateinamens) geladen werden. Anschließend kann das Unterprogramm SETFNA aufgerufen werden. Die Gerätenummer der RS232-Schnittstelle ist 2. Die Sekundäradresse sollte 0 sein. Das erste Byte ist das Kontroll-Register-Byte. Die Bits haben folgende Bedeutung:

Bit 0-3 Baudrate

		3	2	1	0
50	Baud	0	0	0	1
75	Baud	0	0	1	0
110	Baud	0	0	1	1
134.5	Baud	0	1	0	0
150	Baud	0	1	0	1
300	Baud	0	1	1	0
600	Baud	0	1	1	1
1200	Baud	1	0	0	0
1800	Baud	1	0	0	1
2400	Baud	1	0	1	0

Bit 4 nicht verwendet

Bit 5-6 Datenwortlänge

		6	5
8	Bit	0	0
7	Bit	0	1
6	Bit	1	0
5	Bit	1	1

Bit 7 Stopbit

		7
1	Stopb.	0
2	Stopb.	1

Das zweite Byte ist das Befehls-Register-Byte. Es hat folgende Bedeutung:

Bit 0 Handshake

	0
0-3 Leit.	0
X Leit.	1

Bit 1-3 nicht verwendet

Bit 4 Duplex

	4
Full-Duplex	0
Half-Duplex	1

Bit 5-7 Parität

	7	6	5
keine Parit.	*	*	0
ungerade Par.	0	0	1
gearade Par.	0	1	1
Mark Übertr.	1	0	1
Space Übertr.	1	1	1

Sie können zum Beispiel folgende Parameter für Ihren Drucker auswählen:

2 Stopbit
8 Datenbit
300 Baud
3 Handshake Leitungen
Half-Duplex
keine Parität

In diesem Fall müssen die beiden Bytes die Werte $86 und $10 (hexadezimal) enthalten. Nun wird Ihre RS232-Schnittstelle mit diesen Werten arbeiten. Die RS232-Schnittstelle baut zwei Datenpuffer auf, wenn sie eröffnet wird. Jeder hat eine Kapazität von 256 Byte (einer ist für das Empfangen und der andere für das Senden von Daten). Diese zwei Datenpuffer sind am Ende des RAM angeordnet.

Das nächste Programm zeigt, wie man Dateien für Ein und Ausgabe für die RS232-Schnittstelle eröffnet.

```
                *        VORBEREITEN DER RS232-        *
                *        SCHNITTSTELLE ALS EIN-        *
                *        UND AUSGABE                   *
                *        IN BASIC                      *
                *        OPEN1,2,0,CHR$(134)+CHR$(16)  *

                @$FFBA          =SETFPA
                @$FFBD          =SETFNA
                @$FFC0          =OPEN
                * DER DATEINAME RS232 VARIABLE
                @$C100          =FNAME
C100: 86                        #134
C101: 10                        #16
                @2              =FNLEN
                @$C000          =ORG
C000: A901              LDA#    #1
C002: A202              LDX#    #2
C004: A000              LDY#    #0
C006: 20BAFF            JSR     ;SETFPA
C009: A215              LDX#    $0
C00B: A0C0              LDY#    $C1
C00D: A902              LDA#    .FNLEN
C00F: 20BDFF            JSR     ;SETFNA
C012: 20C0FF            JSR     ;OPEN
                * ANWENDERPROGRAMM
                *

ASSEMBLIERUNG BEENDET
```

6

Die Echtzeituhr

Die Echtzeituhr

Ein weiteres Merkmal des C64 ist die eingebaute Echtzeituhr. Diese Echtzeituhr befindet sich in dem Baustein 6526 (CIA = Complex Interface Adapter). Dieser Baustein CIA ist ein neuer Peripheriechip der 65XX-Familie. Die Anfangsadresse dieses Bausteins im C64 ist $DC00.

Die Echtzeituhr ist eine AM/PM-Uhr (AM = ante meridiem = Zeit von 0 - 12 Uhr/ PM = post meridiem = Zeit von 12 bis 24 Uhr) mit einer Genauigkeit von 1/10tel Sekunde. Sie besteht aus vier verschiedenen Registern (1/10s, Sekunden, Minuten und Stunden). Sie werden als BCD-Zahlen ausgegeben, so daß man sie leicht in ASCII-Zahlen umwandeln kann. Zusätzlich zur normalen Tageszeit gibt es noch eine ALARMzeit. Stimmen beide Zeiten überein, gibt der Uhrbaustein einen Interrupt (IRQ) aus. Wollen Sie diese Alarmfunktion benutzen, müssen Sie den softwaremäßigen IRQ-Vektor auf Ihr eigenes Programm zeigen lassen und und das Interrupt Control Register (ICR $DC0F) abfragen. Bei gesetztem Bit 2 stimmen Alarm- und Echtzeituhr überein. Um die Alarmregister anstelle der Echtzeitregister anzuwählen, muß Bit 7 des Control Registers B (CRB $DC0F) gesetzt sein. Folgende Register werden für die Echtzeituhr verwendet:

TOD10TH	$DC08	BIT 0-3 : 1/10tel Sekunde
		BIT 4-7 : immer 0
TODSEC	$DC09	BIT 0-3 : Sekunden
		BIT 4-6 : 10 Sekunden
		BIT 7 : 0
TODMIN	$DC0A	BIT 0-3 : Minuten
		BIT 4-6 : 10 Minuten
		BIT 7 : 0
TODHR	$DC0B	BIT 0-3 : Stunden
		BIT 4 : 10 Stunden
		BIT 5-6 : 0
		BIT 7 : 0 = AM, 1 = PM
ICR	$DC0D	BIT 2 : Alarm = Echtzeit (1)

CRA	$DC0E	BIT 7	: 1 = 50Hz, 0 = 60Hz
CRB	$DC0F	BIT 7	: 1 = Alarm setzen : 0 = Echtzeit setzen

Beim Setzen der Zeit (zuerst die Stunden) hält die Uhr automatisch solange an, bis auch die Speicherstelle TOD10TH gesetzt ist.

Wird die Zeit gelesen, speichert der Uhrenbaustein alle Register in Zwischenregister, die dann gelesen werden können. Erst wenn TOD10TH ebenfalls gelesen wurde, zeigen die Register wieder die aktuelle Zeit an.

Sie sehen also, haben Sie die Echtzeituhr erst einmal gesetzt, dann liefert sie immer die genaue Zeit.

Das folgende Programm setzt die Zeit und geht in eine Abfrageschleife, in der laufend die Zeitregister gelesen und auf dem Bildschirm angezeigt werden.

```
                *        ECHTZEITUHR                    *

                @$FB            =TMP
                @$FD            =AUX
                @$100           =STACK
                @$DC00          =CIA1
                @$DC08          =TOD10TH
                @$DC09          =TODSEC
                @$DC0A          =TODMIN
                @$DC0B          =TODHR
                @$DC0F          =CRB
                @$FFD2          =BSOUT
                @$FFE4          =GET
                @13             =CR
                @$C000          =ORG
                * BILDSCHIRM LOESCHEN
C000: A993              LDA#  #147
C002: 20D2FF            JSR   ;BSOUT
                * AM ODER PM SETZEN
```

```
C005: 20FFC0 =PMSET   JSR   ;PRINT
C008: 0D                    .CR
C009:                       /A ODER P:
C009: 41204F
C00C: 444552
C00F: 2050BA
C012: 202CC1          JSR   ;INPUT
C015: A200            LDX#  #0
C017: C941            CMP#  'A
C019: F006            BEQ   +SETAM
C01B: C950            CMP#  'P
C01D: D0E6            BNE   +PMSET
            * AM --> X=0, PM --> X=128
C01F: A280            LDX#  #128
C021: 86FC   =SETAM   STX.  .TMP+1
            * STUNDEN
C023: 20FFC0 =HRSET   JSR   ;PRINT
C026: 0D                    .CR
C027:                       /STUNDN:
C027: 535455
C02A: 4E444E
C02D: BA
C02E: 2034C1          JSR   ;NINPUT
C031: B0F0            BCS   +HRSET
C033: C902            CMP#  #2
C035: B0EC            BCS   +HRSET
C037: 85FB            STA.  .TMP
C039: 2034C1          JSR   ;NINPUT
C03C: B0E5            BCS   +HRSET
C03E: 2021C1          JSR   ;COMPTMP
C041: 05FC            ORA.  .TMP+1
C043: 8D0BDC          STA   ;TODHR
            * MINUTEN
C046: 20FFC0 =MINSET  JSR   ;PRINT
C049: 0D                    .CR
C04A:                       /MIN  :
C04A: 4D494E
C04D: 20203A
C050: A0
C051: 2034C1          JSR   ;NINPUT
C054: B0F0            BCS   +SECSET
C056: C906            CMP#  #6
C058: B0EC            BCS   +MINSET
C05A: 85FB            STA.  .TMP
```

```
C05C: 2034C1            JSR   ;NINPUT
C05F: B0E5              BCS   +MINSET
C061: 2021C1            JSR   ;COMPTMP
C064: 8D0ADC            STA   ;TODMIN
            * SEKUNDEN
C067: 20FFC0 =SECSET    JSR   ;PRINT
C06A: 0D                      .CR
C06B:                         /SEK  :
C06B: 53454B
C06E: 20203A
C071: A0
C072: 2034C1            JSR   ;NINPUT
C075: B0F0              BCS   +SECSET
C077: C906              CMP#  #6
C079: B0EC              BCS   +SECSET
C07B: 85FB              STA.  .TMP
C07D: 2034C1            JSR   ;NINPUT
C080: B0E5              BCS   +SECSET
C082: 2021C1            JSR   ;COMPTMP
C085: 8D09DC            STA   ;TODSEC
            * 1/10 SEKUNDEN
C088: 20FFC0 =SET10TH JSR     ;PRINT
C08B: 0D                      .CR
C08C:                         /1/10S:
C08C: 312F31
C08F: 30533A
C092: A0
C093: 2034C1            JSR   ;NINPUT
C096: B0F0              BCS   +SET10TH
C098: 8D08DC            STA   ;TOD10TH
            * BILDSCHIRM LOESCHEN
C09B: A993              LDA#  #147
C09D: 20D2FF            JSR   ;BSOUT
C0A0: A913   =LOOP      LDA#  #19
C0A3: 20D2FF            JSR   ;BSOUT
            * AM ODER PM?
C0A5: 2C0BDC            BIT   ;TODHR
C0A8: 1004              BPL   +AM
C0AA: A950              LDA#  'P
C0AC: D002              BNE   +PRINTM
C0AE: A941   =AM        LDA#  'A
C0B0: 20D2FF =PRINTM    JSR   ;BSOUT
C0B3: A94D              LDA#   'M
C0B5: 20D2FF            JSR   ;BSOUT
```

```
C0B8: A920              LDA#  '
C0BA: 20D2FF            JSR   ;BSOUT
              * GIBT STUNDEN AUS
C0BD: AD0BDC            LDA   ;TODHR
C0C0: 297F              AND#  $7F
C0C2: 20EFC0            JSR   ;PRTDIGS
C0C5: A93A              LDA#  ':
C0C7: 20D2FF            JSR   ;BSOUT
              * MINUTEN
C0CA: AD0ADC            LDA   ;TODMIN
C0CD: 20EFC0            JSR   ;PRTDIGS
C0D0: A93A              LDA#  ':
C0D2: 20D2FF            JSR   ;BSOUT
              * SEKUNDEN
C0D5: AD09DC            LDA   ;TODSEC
C0D8: 20EFC0            JSR   ;PRTDIGS
C0DB: 20FFC0            JSR   ;PRINT
C0DE:                         /:
CODE: BA
              * GEBE 1/10 SEKUNDEN AUS
C0DF: AD08DC            LDA   ;TOD10TH
C0E2: 0930              ORA#  '0
C0E4: 20D2FF            JSR   ;BSOUT
C0E7: A930              LDA#  '0
C0E9: 20D2FF            JSR   ;BSOUT
C0EC: 4CA0C0            JMP   ;LOOP
C0EF: 48      =PRTDIGS  PHA
C0F0: 4A                LSRA
C0F1: 4A                LSRA
C0F2: 4A                LSRA
C0F3: 4A                LSRA
C0F4: 20FAC0            JSR   ;DIGOUT
C0F7: 68                PLA
C0F8: 290F              AND#  #15
C0FA: 0930    =DIGOUT   ORA#  '0
C0FC: 4CD2FF            JMP   ;BSOUT
              * ZEICHENAUSGABEPROGRAMM AUS KAPITEL 2
C0FF: 68      =PRINT    PLA
C100: 85FD              STA.  .AUX
C102: 68                PLA
C103: 85FE              STA.  .AUX+1
C105: A200              LDX#  #0
C107: E6FD    =PRINT1   INC.  .AUX
C109: D002              BNE   +PRINT2
```

```
C10B: E6FE              INC.   .AUX+1
C10D: A1FD    =PRINT2   LDA@X  .AUX
C10F: 297F              AND#   $7F
C111: 20D2FF            JSR    ;BSOUT
C114: A200              LDX#   #0
C116: A1FD              LDA@X  AUX
C118: 10ED              BPL    +PRINT1
C11A: A5FE              LDA.   .AUX+1
C11C: 48                PHA
C11D: A5FD              LDA.   .AUX
C11F: 48                PHA
C120: 60                RTS
            * AKKU UND TMP IN EIN BYTE BRINGEN
C121: 06FB    =COMPTMP  ASL.   .TMP
C123: 06FB              ASL.   .TMP
C125: 06FB              ASL.   .TMP
C127: 06FB              ASL.   .TMP
C129: 05FB              ORA.   .TMP
C12B: 60                RTS
            * WARTET AUF TASTENEINGABE, AUSGABE BILDSCHIRM
C12C: 20E4FF  =INPUT    JSR    ;GET
C12F: F0FB              BEQ    +INPUT
C131: 4CD2FF            JMP    ;BSOUT
            * WANDELT EIN ASCII-DIGIT IN EINE BINAERZAHL UM
            * IST DAS DIGIT FALSCH, WIRD CARRY GESETZT
C134: 202CC1  =NINPUT   JSR INPUT
C137: C930              CMP#   '0
C139: 9007              BCC    +NERROR
C13B: C93A              CMP#   ':
C13D: B003              BCS    +NERROR
C13F: 290F              AND#   #15
C141: 60                RTS
C142: 38      =NERROR   SEC
C143: 60                RTS

ASSEMBLIERUNG BEENDET
```

Ihr Rechner verbringt bei diesem Programm den größten Teil der Zeit in einer Dauerschleife. Wir sollten einen besseren Weg finden, die Zeit anzuzeigen, ohne dabei den Rechner zu blockieren. Wir müssen mit einem Interrupt arbeiten. Das folgende Programm kann über BASIC aufgerufen werden.

Mit SYS12*4096 läuft der Dialog an, die Zeit kann gesetzt werden, und danach wird in BASIC zurückgekehrt.

SYS12*4096+3 schaltet den Anzeigemodus ein. In der oberen linken Ecke des Bildschirms erscheint eine Digitaluhr, die die aktuelle Zeit anzeigt.

SYS12*4096+6 schaltet den Anzeigemodus wieder aus, aber die Zeit läuft weiter.

Die Uhr wird erst dann zurückgesetzt, wenn Sie Ihren Rechner aus- und wieder einschalten.

```
            *        ECHTZEITUHR                    *
            *        MIT ANZEIGE DURCH INTERRUPT    *

            @$FB           =TMP
            @$FC           =OLD10TH
            @$FD           =POINT
            @$FD           =AUX
            @$0286         =ACTCOL
            @$0314         =IRQVECT
            @$0400         =SCREEN
            @$D800         =COLOR
            @$DC00         =CIA1
            @$DC08         =TOD10TH
            @$DC09         =TODSEC
            @$DC0A         =TODMIN
            @$DC0B         =TODHR
            @$DC0F         =CRB
            @$FFD2         =BSOUT
            @$FFE4         =GET
            @$EA           =OLDIRQH
            @$31           =OLDIRQL
            @$EA31         =OLDIRQ
            @13            =CR
            @19            =HOME
            @147           =CLS
            @$C000         =ORG
            * ZEIT SETZEN SYS12*4096
C000: 4C09C0          JMP   ;SET
```

```
            * ZEIT FUER ANZEIGE SYS 12*4096+3
C003: 4C2CC0          JMP   ;INSTALL

            * ANZEIGE EIN SYS 12*4096+6
C006: 4C39C0          JMP   ;LEAVE
            * IRQVECTOR ZEIGT AUF EIGENEN TREIBER
C009: A993  =SET      LDA#  #CLS
C00B: 20D2FF          JSR   ;BSOUT
            * CRB SETZEN FUER ECHTZEIT
C00E: AD0FDC          LDA   ;CRB
C011: 297F            AND#  #127
C013: 8D0FDC          STA   ;CRB
            * MELDUNG AUSGEBEN, ECHTZEIT SETZEN
C016: 2078C1          JSR   ;PRINT
C019: 0D                    .CR
C01A:                       /TAG.ZEIT SETZEN
C01A: 544147
C01D: 2E5A45
C020: 495420
C023: 534554
C026: 5A45CE
C029: 4C46C0          JMP   ;CLOCK
C02C: 78    =INSTALL SEI
C02D: A900            LDA#  .OWNIRQL
C02F: 8D1403          STA   ;IRQVECT
C032: A9C1            LDA#  .OWNIRQH
C034: 8D1503          STA   ;IRQVECT+1
C037: 58              CLI
C038: 60              RTS
C039: 78    =LEAVE    SEI
C03A: A931            LDA#  .OLDIRQL
C03C: 8D1403          STA   ;IRQVECT
C03F: A9EA            LDA#  .OLDIRQH
C041: 8D1503          STA   ;IRQVECT+1
C044: 58              CLI
C045: 60              RTS
            * AM --> X=0, PM --> X=128
C046: 2078C1 =SETCLCK JSR   ;PRINT
C049: OD                    .CR
C04A:                       /A ODER P:
C04A: 41204F
C04D: 444552
C050: 2050BA
C053: 20A5C1          JSR   ;INPUT
```

```
C056: A200              LDX#  #0
C058: C941              CMP#  'A
C05A: F006              BEQ   +SETAM
C05C: C950              CMP#  'P
C05E: D0E6              BNE   +SETCLCK
C060: A280              LDX#  #128
C062: 86FC    =SETAM    STX.  .TMP+1
              * STUNDEN
C064: 2078C1  =HRSET    JSR   ;PRINT
C067: 0D                      .CR
C068:                         /STUNDN:
C068: 535455
C06B: 4E444E
C06E: BA
C06F: 20ADC1            JSR   ;NINPUT
C072: B0F0              BCS   +HRSET
C074: C902              CMP#  #2
C076: B0EC              BCS   +HRSET
C078: 85FB              STA.  .TMP
C07A: 20ADC1            JSR   ;NINPUT
C07D: B0E5              BCS   +HRSET
C07F: 209AC1            JSR   ;COMPTMP
C082: 05FC              ORA.  .TMP+1
C084: 8D0BDC            STA   ;TODHR
              * MINUTEN
C087: 2078C1  =MINSET   JSR   ;PRINT
C08A: 0D                      .CR
C08B:                         /MIN  :
C08B: 4D494E
C08E: 20203A
C091: A0
C092: 20ADC1            JSR   ;NINPUT
C095: B0F0              BCS   +SECSET
C097: C906              CMP#  #6
C099: B0EC              BCS   +MINSET
C09B: 85FB              STA.  .TMP
C09D: 20ADC1            JSR   ;NINPUT
C0A0: B0E5              BCS   +MINSET
C0A2: 209AC1            JSR   ;COMPTMP
C0A5: 8D0ADC            STA   ;TODMIN
              * SEKUNDEN
C0A8: 2078C1  =SECSET   JSR   ;PRINT
C0AB: 0D                      .CR
C0AC:                         /SEK  :
```

```
C0AC: 53454B
C0AF: 20203A
C0B2: A0
C0B3: 20ADC1          JSR   ;NINPUT
C0B6: B0F0            BCS   +SECSET
C0B8: C906            CMP#  #6
C0BA: B0EC            BCS   +SECSET
C0BC: 85FB            STA.  .TMP
C0BE: 20ADC1          JSR   ;NINPUT
C0C1: B0E5            BCS   +SECSET
C0C3: 209AC1          JSR   ;COMPTMP
C0C6: 8D09DC          STA   ;TODSEC
            * 1/10 SEKUNDEN (IMMER 0)
C0C9: A900            LDA#  #0
C0CB: 8D08DC          STA   ;TOD10TH
C0CE: 60              RTS
            * IRQ TREIBER: VERAENDERN SICH 1/10 S, WIRD NEUE
            * ZEIT ANGE ZEIGT, SONST SPRUNG IN DEN OS TREIBER
            @$C1              =OWNIRQH
            @$0               =OWNIRQL
            @$C100            =OWNIRQ
C100: AD08DC          LDA   ;TOD10TH
C103: C5FC            CMP.  .OLD10TH
C105: F005            BEQ   +NORMIRQ
C107: 85FC            STA.  .OLD10TH
C109: 200FC1          JSR   ;SHOWTIM
C10C: 4C31EA =NORMIRQ JMP   ;OLDIRQ
            * ZEITANZEIGE AUF DEM BILDSCHIRM
C10F: 78     =SHOWTIM SEI
C110: A900            LDA#  #0
C112: 85FD            STA.  .POINT
            * AM ODER PM AUSGABE
C114: 2C0BDC          BIT   ;TODHR
C117: 1004            BPL   +AM
C119: A910            LDA#  #16
C11B: D002            BNE   +PRINTM
C11D: A901   =AM      LDA   #1
C11F: 2069C1 =PRINTM  JSR   ;ZOUTPUT
C122: A920            LDA#  '
C124: 2069C1          JSR   ;ZOUTPUT
            * STUNDENAUSGABE
C127: AD0BDC          LDA   ;TODHR
C12A: 297F            AND#  $7F
C12C: 2059C1          JSR   ;PRTDIGS
```

```
C12F: A93A              LDA#  ':
C131: 2069C1            JSR   ;ZOUTPUT
                * MINUTEN
C134: AD0ADC            LDA   ;TODMIN
C137: 2059C1            JSR   ;PRTDIGS
C13A: A93A              LDA#  ':
C13C: 2069C1            JSR   ;ZOUTPUT
                * SEKUNDEN
C13F: AD09DC            LDA   ;TODSEC
C142: 2059C1            JSR   ;PRTDIGS
C145: A93A              LDA#  ':
C147: 2069C1            JSR   ;ZOUTPUT
                * GEBE 1/10 SEKUNDEN AUS
C14A: AD08DC            LDA   ;TOD10TH
C14D: 0930              ORA#  '0
C14F: 2069C1            JSR   ;ZOUTPUT
C152: A930              LDA#  '0
C154: 2069C1            JSR   ;ZOUTPUT
C157: 58                CLI
C158: 60                RTS
C159: 48        =PRTDIGS PHA
C15A: 4A                LSRA
C15B: 4A                LSRA
C15C: 4A                LSRA
C15D: 4A                LSRA
C15E: 2064C1            JSR   ;DIGOUT
C161: 68                PLA
C162: 290F              AND#  #15
C164: 0930      =DIGOUT ORA#  '0
C166: 4C69C1            JMP   ;ZOUTPUT
                * ZEICHEN AUF DEN SCHIRM
C169: A6FD      =ZOUTPUT LDX. .POINT
C16B: 9D0004            STAX  .SCREEN
C16E: AD8602            LDA   ;ACTCOL
C171: 9D00D8            STAX  .COLOR
C174: E8                INX
C175: 86FD              STX.  .POINT
C177: 60                RTS
                * ZEICHENAUSGABEPROGRAMM AUS KAPITEL 2
C178: 68        =PRINT  PLA
C179: 85FD              STA.  .AUX
C17B: 68                PLA
C17C: 85FE              STA.  .AUX+1
C17E: A200              LDX#  #0
```

```
C180: E6FD    =PRINT1  INC.   .AUX
C182: D002             BNE    +PRINT2
C184: E6FE             INC.   .AUX+1
C186: A1FD    =PRINT2  LDA@X  .AUX
C188: 297F             AND#   $7F
C18A: 20D2FF           JSR    ;BSOUT
C18D: A200             LDX#   #0
C18F: A1FD             LDA@X  .AUX
C191: 10ED             BPL    +PRINT1
C193: A5FE             LDA.   .AUX+1
C195: 48               PHA
C196: A5FD             LDA.   .AUX
C198: 48               PHA
C199: 60               RTS
              * AKKU UND TMP IN EIN BYTE BRINGEN
C19A: 06FB    =COMPTMP ASL.   .TMP
C19C: 06FB             ASL.   .TMP
C19E: 06FB             ASL.   .TMP
C1A0: 06FB             ASL.   .TMP
C1A2: 05FB             ORA.   .TMP
C1A4: 60               RTS
              * WARTET AUF TASTENEINGABE, AUSGABE BILDSCHIRM
C1A5: 20E4FF  =INPUT   JSR    ;GET
C1A8: F0FB             BEQ    +INPUT
C1AA: 4CD2FF           JMP    ;BSOUT
              * WANDELT EIN ASCII-DIGIT IN EINE BINAERZAHL UM
              * IST DAS DIGIT FALSCH, WIRD CARRY GESETZT
C1AD: 20A5C1  =NINPUT  JSR    ;INPUT
C1B0: C930             CMP#   '0
C1B2: 9007             BCC    +NERROR
C1B4: C93A             CMP#   ':
C1B6: B003             BCS    +NERROR
C1B8: 290F             AND#   #15
C1BA: 60               RTS
C1BB: 38      =NERROR  SEC
C1BC: 60               RTS

ASSEMBLIERUNG BEENDET
```

7

Wie man neue BASIC-Befehle einfügt

Wie man neue BASIC-Befehle einfügt

In BASIC können Sie Ihre eigenen maschinengeschriebenen Unterprogramme mit dem SYS-Befehl aufrufen. Es ist auch möglich, Ihre eigenen Befehle innerhalb des BASIC-Interpreters unterzubringen.

Zuerst sehen wir uns an, wie der SYS-Befehl verwendet wird, um Ihre eigenen Kommandos in BASIC auszuführen. Wollen Sie ein Unterprogramm vom BASIC aus aufrufen, dann müssen Sie es in einem bestimmten Speicherbereich ablegen. Dieser geschützte Speicherbereich beginnt bei $C000 (12*4096) und endet bei $CFFF. Alle unsere Beispiele laufen in diesem Bereich.

Manchmal möchten Sie an Ihr Unterprogramm von BASIC aus Parameter übergeben. Wir diskutieren einige dieser Unterprogramme und zeigen ein Programmbeispiel.

Name : CHECKCOM
Adresse : $AEFD

Dieses Unterprogramm sucht nach einem Komma (',') und überspringt es. Der BASIC-Zeiger zeigt nun unmittelbar auf die Stelle hinter dem Komma. Wird kein Komma gefunden, unterbricht BASIC und gibt eine Fehlermeldung aus.

Name : GETCOORD
Adresse : $B7EB

Dieses Unterprogramm holt zwei Zahlen, die durch ein Komma getrennt sein müssen. Die erste Zahl (Ganzzahl ohne Vorzeichen) ist eine 16 Bit Zahl und wird auf den Speicherplätzen $14 (LSB) und $15 (MSB) abgelegt. Die zweite Zahl ist ein Byte-Ausdruck. Das heißt, daß es sich um eine Zahl im Bereich zwischen 0 und 255 handelt. Der Wert wird im X-Register übergeben. Sind die Werte der zwei Zahlen außerhalb des zulässigen Bereichs, wird eine Fehlermeldung ausgegeben und BASIC unterbricht. Auch ein fehlendes Komma stoppt BASIC.

Wie man neue BASIC-Befehle einfügt

Name : GETBYTE
Adresse : $B79E

GETBYTE holt einen Byte-Ausdruck (0 bis 255) und überträgt ihn ins X-Register. Ist der Ausdruck außerhalb des zulässigen Bereichs, unterbricht BASIC und gibt eine Fehlermeldung aus.

Name : GETPARAM
Adresse : $E1D4

GETPARAM holt den Dateinamen und die Gerätenummer. Wenn die Dateispezifikation falsch ist, erscheint eine Fehlermeldung.

Das Programm im nächsten Kapitel verwendet diese Unterprogramme.

Anstelle des SYS-Befehls können Sie Ihre eigenen Schlüsselworte benutzen. Dazu müssen Sie einen Eingriff in den BASIC-Interpreter vornehmen. Das nächste Bild zeigt uns einen Teil der Schleife:

```
A7E1  6C0803 JMP ($0308)   ; zeigt auf die nächste Instruktion
A7E4  207300 JSR $0073     ; CHRGET holt das nächste Zeichen
A7E7  20EDA7 JSR $A7ED     ; führt BASIC-Anweisung aus
A7EA  4CAEA7 JMP $A7AE     ; springt an den Anfang der Schleife
```

Sie können also den Zeiger ($0308, $0309) ändern und auf Ihren eigenen 'Interpreter' zeigen lassen. Dieses Entschlüsselungsprogramm prüft auf Ihre eigenen Schlüsselworte. Sollten Sie das nicht wünschen, springen Sie einfach nach $A7E7 und der Interpreter wird normal weitermachen. Sie lassen Ihr eigenes Programm ablaufen und springen dann nach $A7AE, an den Anfang der Interpreter-Schleife. Gewöhnlich stellt man den eigenen Befehlen zur besseren Unterscheidung ein Symbol vor, wie '@,'! usw. So müssen Sie nur auf dieses Symbol abfragen, in Ihrerer eigenen Befehlstabelle nachsehen und den Befehl, wenn er gültig war, ausführen. Kapitel 9 beschreibt ein Programm, das zusätzliche Befehle verwendet.

Näheres über die Unterprogramme in BASIC können Sie den Handbüchern über C64-Speicheraufteilung und das ROM-Listing entnehmen.

8

Hochauflösende Grafik

Hochauflösende Grafik

Das folgende Programm ermöglicht Ihnen das Arbeiten im hochauflösenden Grafikmodus Ihres C64. Das heißt, Sie können mit dieser Programmhilfe in BASIC sehr schnell Bildpunkte setzen oder löschen. Außerdem dient es als Beispiel für die Verwendung von SYS-Befehlen bei der Ausführung Ihrere eigenen BASIC-Programme. Das Programm enthält eine Sprungtabelle. Sie müssen nur jeweils drei hinzuaddieren, damit Sie zur Anfangsadresse des nächsten Befehls gelangen. SYS12*4096 initialisiert den Bildschirm. SYS12*4096+3 löscht den Bildschirm und SYS12*4096+6,5 schaltet die Hintergrundfarbe 5 (grün) ein. Sie können den Objekt-Code dieses Programmes mit einem Monitor abspeichern und es in BASIC mit dem LOAD-Befehl laden. Dateiname, Gerätenummer und Sekundäradresse (= 1) müssen im Befehl aufgeführt sein. Das Programm wird an die Adresse $C000 geladen. Danach müssen Sie NEW eingeben. Nun können Sie Ihr eigenes BASIC-Programm laden. Mit dem Programm 'Hochauflösende Grafik' können Sie den Inhalt des Bildschirms auf Kassette oder Diskette abspeichern und wieder lesen. Das Programm verwendet Unterprogramme, die in vorhergehenden Kapiteln bereits besprochen wurden. Als Beispiel sollten Sie einmal versuchen, das Programm mit einem Zeichenbefehl (drawto) zu erweitern, damit Sie schnell Linien ziehen können.

Das BASIC-Programm ist ein Test für das anschließende Assemblerprogramm.

```
5 REM TESTDEMO HOCHAUFLÖSENDE GRAFIK
10 INPUT"X1=WERT ";X1
20 INPUT"Y1=WERT ";Y1
30 INPUT"X2=WERT ";X2
40 INPUT"Y2=WERT ";Y2
45 SYS12*4096:SYS12*4096+3
55 DX=X2-X1:DY=Y2-Y1
57 IF ABS(DY)<ABS(DX) THEN 64
58 FOR YL=Y1 TO Y2 STEP SGN(DY)
60 SYS12*4096+9,DX/DY*YL+X1,YL
62 NEXT YL
63 GOTO 300
64 FOR YL=X1 TO X2 STEP SGN(DX)
65 SYS12*4096+9,XL,DY/DX*XL+Y1
```

```
68 NEXT XL
300 IF ABS(DY)<ABS(DX) THEN 350
310 FOR YL=Y1 TO Y2 STEP SGN(DY)
320 SYS12*4096+12,DX/DY*YL+X1,YL
330 NEXT YL
340 GOTO 380
350 FOR XL=X1 TO X2 STEP SGN(DX)
360 SYS12*4096+12,XL,DY/DX*XL+Y1
370 NEXT XL
380 GOTO 57
```

```
            *        HILFSPROGRAMM ZUR ER-        *
            *        STELLUNG HOCHAUFLOE-         *
            *        SENDER GRAFIKEN              *

           @$14            =XCOORD
           @$B9            =DECADR
           @$FD            =TEMP
           @$FD            =ADDRESS
           @$4             =COLORLH
           @$8             =COLORHH
           @$20            =GRAPHLH
           @$0             =GRAPHLL
           @$40            =GRAPHHH
           @$0             =GRAPHHL
           @$AEFD          =CHCKCOM
           @$B79E          =GETBYTE
           @$B7EB          =GETCOOR
           @$E1D4          =GETPARM
           @$FFD2          =BSOUT
           @$FFD5          =LOAD
           @$FFD8          =SAVE
           @$D000          =VIDEO
           @255            =FALSE
           @0              =TRUE
           @147            =CLS
           @$C000          =ORG
           * INITIALISIERUNG HOCHAUFLOESENDE GRAFIK SYS12*4096
C000: 4C18C0          JMP   ;INIT
           * BILDSCHIRM LOESCHEN SYS12*4096+3
C003: 4C33C0          JMP   ;CLEAR
           * HINTERGRUNDFARBE SYS12*4096+6,COLOR
C006: 4C4AC0          JMP   ;COLOR
```

```
                * PLOT X,Y (0 <= X < 320); (0 <= Y < 200)
                * SYS12*4096+9,X,Y
C009: 4C6BC0            JMP   ;SET
                * X,Y LOESCHEN SYS12*4096+12,X,Y
C00C: 4C67C0            JMP   ;RESET
                * GRAFIK AUS, ZURUECK IN NORMALMODUS, SYS12*4096+15
C00F: 4CD8C0            JMP   ;SWCHOFF
                * GRAFIK ABSPEICHERN SYS12*4096+18,"NAME",GERAET
C012: 4CE9C0            JMP   ;SCREENS
                * GRAFIK LADEN SYS12*4096+21,"NAME",GERAET
C015: 4C00C1            JMP   ;SCREENL
C018: AD11D0 =INIT      LDA   ;VIDEO+17
C01B: 8D52C1            STA   ;SCRATCH+1
C01E: AD18D0            LDA   ;VIDEO+24
C021: 8D51C1            STA   ;SCRATCH
C024: A93B              LDA#  #59
C026: 8D11D0            STA   ;VIDEO+17
C029: A918              LDA#  #24
C02B: 8D18D0            STA   ;VIDEO+24
C02E: A210              LDX#  #16
C030: 4C50C0            JMP   ;COLOR1
C033: A000   =CLEAR     LDY#  #0
C035: A920              LDA#  #GRAPHLH
C037: 84FD              STY.  .TEMP
C039: 85FE              STA.  .TEMP+1
C03B: 98     =CLEAR1    TYA
C03C: 91FD   =CLEAR2    STA@Y  .TEMP
C03E: C8                INY
C03F: D0FB              BNE   +CLEAR2
C041: E6FE              INC.  .TEMP+1
C043: A5FE              LDA.  .TEMP+1
C045: C940              CMP#  .GRAPHHH
C047: D0F2              BNE   ;CLEAR1
C049: 60                RTS
C04A: 20FDAE =COLOR     JSR   ;CHCKCOM
C04D: 209EB7            JSR   ;GETBYTE
C050: A000   =COLOR1    LDY#  #0
C052: A904              LDA#  .COLORLH
C054: 84FD              STY.  .TEMP
C056: 85FE              STA.  .TEMP+1
C058: 8A     =COLOR2    TXA
C059: 91FD   =COLOR3    STA@Y .TEMP
C05B: C8                INY
C05C: DOFB              BNE   +COLOR3
```

```
C05E: E6FE              INC.   .TEMP+1
C060: A5FE              LDA.   .TEMP+1
C062: C908              CMP#   .COLORHH
C064: D0F2              BNE    +COLOR2
C066: 60      =OUTRANG  RTS
C067: A9FF    =RESET    LDA#   .FALSE
C069: D002              BNE    +SET1
C06B: A900    =SET      LDA#   #TRUE
C06D: 8D53C1  =SET1     STA    ;RSFLG
C070: 20FDAE            JSR    ;CHCKCOM
C073; 20EBB7            JSR    ;GETCOOR
C076: E0C8              CPX#   #200
C078: B0EC              BCS    +OUTRANG
C07A: A514              LDA.   .XCOORD
C07C: C940              CMP#   #64
C07E: A515              LDA.   .XCOORD+1
C080: E901              SBC#   #1
C082: B0E2              BCS    +OUTRANG
C084: 8A                TXA
C085: 4A                LSRA
C086: 4A                LSRA
C087: 4A                LSRA
C088: 0A                ASLA
C089: A8                TAY
C08A: B90FC1            LDAY   .MUL320
C08D: 85FD              STA.   .ADDRESS
C08F: B910C1            LDAY   .MUL32+1
C092: 85FE              STA    ;ADDRESS+1
C094: 8A                TXA
C095: 2907              AND#   #7
C097: 18                CLC
C098: 65FD              ADC.   .ADDRESS
C09A: 85FD              STA.   .ADDRESS
C09C: A5FE              LDA.   .ADDRESS+1
C09E: 6900              ADC#   #0
C0A0: 85FE              STA.   .ADDRESS+1
C0A2: A514              LDA.   .XCOORD
C0A4: 2907              AND#   #7
C0A6: A8                TAY
C0A7: A514              LDA.   .XCOORD
C0A9: 29F8              AND#   $F8
C0AB: 18                CLC
C0AC: 65FD              ADC.   .ADDRESS
C0AE: 85FD              STA.   .ADDRESS
```

```
C0B0: A5FE              LDA.   .ADDRESS+1
C0B2: 6515              ADC.   .XCOORD+1
C0B4: 85FE              STA.   .ADDRESS+1
C0B6: A5FD              LDA.   .ADDRESS
C0B8: 18                CLC
C0B9: 6900              ADC#   #GRAPHLL
C0BB: 85FD              STA.   .ADDRESS
C0BD: A5FE              LDA.   .ADDRESS+1
C0BF: 6920              ADC#   .GRAPHLH
C0C1: 85FE              STA.   .ADDRESS+1
C0C3: A200              LDX#   #0
C0C5: A1FD              LDA@X  .ADDRESS
C0C7: 2C53C1            BIT    ;RSFLG
C0CA: 1006              BPL    +SET2
C0CC: 3949C1            ANDY   .ANDMASK
C0CF: 4CD5C1            JMP    ;SET3
C0D2: 1941C1  =SET2     ORAY   .ORMASK
C0D5: 81FD    =SET3     STA@X  .ADDRESS
C0D7: 60                RTS
C0D8: AD52C1  =SWCHOFF  LDA    ;SCRATCH+1
C0DB: 8D11D0            STA    ;VIDEO+17
C0DE: AD51C1            LDA    ;SCRATCH
C0E1: 8D18D0            STA    ;VIDEO+24
C0E4: A993              LDA#   .CLS
C0E6: 4CD2FF            JMP    ;BSOUT
C0E9: 20FDAE  =SCREENS  JSR    ;CHCKCOM
C0EC: 20D4E1            JSR    ;GETPARM
C0EF: A200              LDX#   .GRAPHHL
C0F1: A040              LDY#   .GRAPHHH
C0F3: A900              LDA#   .GRAPHLL
C0F5: 85FD              STA.   .TEMP
C0F7: A920              LDA#   .GRAPHLH
C0F9: 85FE              STA.   .TEMP+1
C0FB: A9FD              LDA#   .TEMP
C0FD: 4CD8FF            JMP    ;SAVE
C100: 20FDAE  =SCREENL  JSR    ;CHCKCOM
C103: 20D4E1            JSR    ;GETPARM
C106: A961              LDA#   #97
C108: 85B9              STA.   .SECADR
C10A: A900              LDA#   #0
C10C: 4CD5FF            JMP    ;LOAD
              * MULTIPLIKATIONSTABELLE (* 320)
C10F: 0000    =MUL320          ;#0
C111: 4001                     ;#320
```

```
C113: 8002                    ;#640
C115: C003                    ;#960
C117: 0005                    ;#1280
C119: 4006                    ;#1600
C11B: 8007                    ;#1920
C11D: C008                    ;#2240
C11F: 000A                    ;#2560
C121: 400B                    ;#2880
C123: 800C                    ;#3200
C125: C00D                    ;#3520
C127: 000F                    ;#3840
C129: 4010                    ;#4160
C12B: 8011                    ;#4480
C12D: C012                    ;#4800
C12F: 0014                    ;#5120
C131: 4015                    ;#5440
C133: 8016                    ;#5760
C135: C017                    ;#6080
C137: 0019                    ;#6400
C139: 401A                    ;#6720
C13B: 801B                    ;#7040
C13D: C01C                    ;#7360
C13F: 001E                    ;#7680
            * MASKE  BITSETZEN FUER DAS ANGEWAEHLTE BYTE
C141: 80    =ORMASK           $80
C142: 40                      $40
C143: 20                      $20
C144: 10                      $10
C145: 08                      $8
C146: 04                      $4
C147: 02                      $2
C148: 01                      $1
            * MASKE BITRUECKSETZEN FUER DAS ANGEWAEHLTE BYTE
C149: 7F    =ANDMASK          $7F
C14A: BF                      $BF
C14B: DF                      $DF
C14C: EF                      $EF
C14D: F7                      $F7
C14E: FB                      $FB
C14F: FD                      $FD
C150: FE                      $FE
C151: 0000  =SCRATCH          ;#0
C153: 00    =RSFLG            #0
C154: 00    =YCOORD           #0
```

9

Hilfsprogramm zur Diskettenbedienung

Hilfsprogramm zur Diskettenbedienung

Das hier beschriebene Programm erweitert BASIC um folgende Disketten-Befehle:

DLOAD
Arbeitet wie die bekannte LOAD-Anweisung, aber die Gerätenummer ist immer 8. Der Dateiname kann mit der Sekundäradresse gesetzt werden. Einzelheiten lesen Sie bitte im Commodore-Handbuch nach.

DSAVE
Der SAVE-Befehl für Gerät 8, wie DLOAD.

DPRINT
Dieser Befehl kann Befehle mit einer einzigen Anweisung , wie NEW, COPY, RENAME, unmittelbar zur Diskette senden. Er entspricht der folgenden BASIC-Sequenz:

```
OPEN15,8,15,"<Befehl>":CLOSE15
```

DERROR
Zeigt den Zustand der Diskette an, ersetzt das folgende Programm:

```
10 OPEN15,8,15
20 INPUT#15,A$,B$,C$,D$
30 PRINTA$,B$,C$,D$
40 CLOSE15
```

Die DERROR-Anweisung gibt die Fehlernummer, Fehlermeldung, Spurnummer und Sektornummer aus.

DLIST
Dieser Befehl zeigt das Inhaltsverzeichnis der Diskette, dabei wird das im Speicher stehende BASIC-Programm nicht zerstört.

Das Hilfsprogramm muß mit der Sekundäradresse = 1 geladen werden (absolutes Laden). Danach geben Sie NEW und SYS12*4096 ein. Nun können Sie die neuen Befehle benutzen.

SYS12*4096 ruft ein Programm auf, das in die Interpreterschleife eingreift (der Vektor auf $0308-$0309 wird geändert).

```
                *          HILFSPROGRAMM ZUR          *
                *          DISKETTEN BEDIENUNG        *

                @$A              =FLAG
                @$7A             =BASICP
                @$90             =ST
                @$B7             =FNLENG
                @$B9             =SECADR
                @$BA             =DEVNUM
                @$BB             =FNADRES
                @$FB             =TMP
                @$0073           =CHRGET
                @$A7E7           =EXECUTE
                @$A7AE           =INTERP
                @$BDCD           =LNPRT
                @$E200           =PARGET
                @$E257           =GETNAME
                @$E206           =NEXTQT
                @$E0F9           =GIVERR
                @$F3D5           =SENDFNA
                @$F642           =CLOSEFI
                @$FF96           =SENDSEC
                @$FFA5           =IECINP
                @$FFAB           =UNTALK
                @$FFB4           =IECTALK
                @$FFBA           =SETFPA
                @$FFBD           =SETFNA
                @$FFC0           =OPEN
                @$FFC3           =CLOSE
                @$FFD2           =BSOUT
                @$E16F           =BLOAD
                @$E159           =BSAVE
                @147             =LOADTOK
                @148             =SAVETOK
                @149             =VERITOK
                @153             =PRINTTO
                @176             =ORTOKEN
                @155             =LISTTOK
                @13              =CR
                @$C000           =ORG
                * NEUE BEFEHLE HINZUFUEGEN MIT SYS12*4096
C000: A90B            LDA#  $B
C002: 8D0803          STA   ;$0308
C005: A9C0            LDA   $C0
```

```
C007: 8D0903          STA   ;$0309
C00A: 60              RTS
            * DEKODIERER, ALLE NEUEN BEFEHLE BEGINNEN MIT 'D
C00B: 207300 =DECODE  JSR   ;CHRGET
C00E: C944            CMP#  'D
C010: F003            BEQ   +FOUND
C012: 4CE7A7          JMP   ;EXECUT
C015: A001   =FOUND   LDY#  #1
C017: B17A            LDA@Y .BASICP
            * DLOAD ?
C019: C993            CMP#  .LOADTOK
C01B: F031            BEQ   +DLOAD
            * DSAVE ?
C01D: C994            CMP#  .SAVETOK
C01F: F040            BEQ   +DSAVE
            * DVERIFY ?
C021: C995            CMP#  .VERITOK
C023: F026            BEQ   +DVERIFY
            * DPRINT?
C025: C999            CMP#  .PRINTTO
C027: F01C            BEQ   +DOPRINT
            * DLIST ?
C029: C99B            CMP#  .LISTTOK
C02B: F01B            BEQ   +DODLIST
            * DERROR
C02D: A200            LDX#  #0
C02F: B17A   =COMPLP  LDA@Y .BASICP
C031: DD64C1          CMPX  .DERRTAB
C034: D008            BNE   NDERR
C036: C8              INY
C037: E8              INX
C038: E004            CPX#  #4
C03A: 90F3            BCC   +COMPLP
C03C: B051            BCS   +DERROR
C03E: A000   =NDERR   LDY#  #0
C040: B17A            LDA@Y .BASICP
C042: 4CE7A7          JMP   ;EXECUT
C045: 4CC5C0 =DOPRINT JMP   ;DPRINT
C048: 4CF4C0 =DODLIST JMP   ;DLIST
            * DVERIFY VERGLEICHT EIN PROGRAMM IM SPEICHER MIT
            * MIT EINEM DISKETTENPROGRAMM
            * DVERIFY"NAME",SA (SA IST OPTIONAL)
C04B: A901   =DVERIFY LDA#  #1
C04D: 2C              $2C
```

```
                * DLOAD LAEDT EIN PROGRAMM VON DER DISKETTE IN DEN
                * ARBEITSSPEICHER: DLOAD"NAME",SA (SA IST OPTIONAL)
C04D: A900      =DLOAD   LDA#  #0
C050: 850A               STA.  .FLAG
C052: 207300             JSR   ;CHRGET
C055: 207300             JSR   ;CHRGET
C058: 207300             JSR   ;CHRGET
C05B: 206FE1             JSR   ;BLOAD
C05E: 4CAEA7             JMP   ;INTERP
                * DSAVE SPEICHERT EIN PROGRAMM AUS DEM ARBEITSSPEI-
                * CHER AUF DISKETTE: DSAVE"NAME",SA (SA OPTIONAL)
C061: 207300    =DSAVE   JSR   ;CHRGET
C064: 207300             JSR   ;CHRGET
C067: 2070C0             JSR   ;GETPAR
C06A: 2059E1             JSR   ;BSAVE
C06D: 4CAEA7             JMP   ;INTERP
C070: A900      =GETPAR  LDA#  #0
C072: 20BDFF             JSR   ;SETFNA
C075: A208               LDX#  #8
C077: A000               LDY#  #0
C079: 20BAFF             JSR   ;SETFPA
C07C: 2006E2             JSR   ;NEXTQ
C07F: 2057E2             JSR   ;GETNAME
C082: 2006E2             JSR   ;NEXTQ
C085: 2000E2             JSR   ;PARGET
C088: 8A                 TXA
C089: A8                 TAY
C08A: A208               LDX#  #8
C08C: 4CBAFF             JMP   ;SETFPA
                * DERROR LIESTDEN FEHLER KANAL (15) DER DISKETTE
                * UND GIBT FEHLERNUMMER, SPUR UND SEKTOR AUS
C08F: A57A      =DERROR  LDA.  .BASICP
C091: 18                 CLC
C092: 6905               ADC#  #5
C094: 857A               STA.  .BASICP
C096: A57B               LDA.  .BASICP+1
C098: 6900               ADC#  #0
C09A: 857B               STA.  .BASICP+1
C09C: A900               LDA#  #0
C09E: 8590               STA.  .ST
C0A0: A908               LDA#  #8
C0A2: 85BA               STA.  .DEVNUM
C0A4: 20B4FF             JSR   ;IECTALK
C0A7: A96F               LDA#  #111
```

```
C0A9: 85B9              STA.  .SECADR
C0AB: 2096FF            JSR   ;SENDSEC
C0AE: A490    =DERRLP   LDY.  .ST
C0B0: D00A              BNE   +DERR4
C0B2: 20A5FF            JSR   ;IECINP
C0B5: 20D2FF            JSR   ;BSOUT
C0B8: C90D              CMP#  .CR
C0BA: D0F2              BNE   +DERRLP
C0BC: 20ABFF  =DERR4    JSR   ;UNTALK
C0BF: 4CAEA7            JMP   ;INTERP
C0C2: 4CF9E0  =DERRERR  JMP   ;GIVERR
              * DPRINT SENDET BEFEHLE UEBER DEN BEFEHLSKANAL 15
              * SEHEN SIE BITTE IM DISKETTENHANDBUCH NACH
C0C5: 207300  =DPRINT   JSR   ;CHRGET
C0C8: 207300            JSR   ;CHRGET
C0CB: A90F              LDA#  #15
C0CD: 20C3FF            JSR   ;CLOSE
C0D0: 20E0C0            JSR   ;GETFPAR
C0D3: 20C0FF            JSR   ;OPEN
C0D6: B0EA              BCS   ;DERRERR
C0D8: A90F              LDA#  #15
C0DA: 20C3FF            JSR   ;CLOSE
C0DD: 4CAEA7            JMP   ;INTERP
C0E0: A900    =GETFPAR  LDA#  #0
C0E2: 20BDFF            JSR   ;SETFNA
C0E5: A90F              LDA#  #15
C0E7: A8                TAY
C0E8: A208              LDX#  #8
C0EA: 20BAFF            JSR   ;SETFPA
C0ED: 2006E2            JSR   ;NEXTQ
C0F0: 2057E2            JSR   ;GETNAME
C0F3: 60                RTS
              * DLIST GIBT DAS INHALTSVERZEICHNIS DER DISKETTE
              * AUS OHNE DEN SPEICHER ZU ZERSTOEREN
C0F4: 207300  =DLIST    JSR   ;CHRGET
C0F7: 207300            JSR   ;CHRGET
C0FA: A900              LDA#  #0
C0FC: 8590              STA.  .ST
C0FE: A924              LDA#  '$
C100: 85FB              STA.  .TMP
C102: A9FB              LDA#  #TMP
C104: 85BB              STA.  .FNADRES
C106: A900              LDA#  #0
C108: 85BC              STA.  FNADRES+1
```

```
C10A: A901              LDA#  #1
C10C: 85B7              STA.  .FNLENG
C10E: A908              LDA#  #8
C110: 85BA              STA.  .DEVNUM
C112: A960              LDA#  #96
C114: 85B9              STA.  .SECADR
C116: 20D5F3            JSR   ;SENDFNA
C119: A5BA              LDA.  .DEVNUM
C11B: 20B4FF            JSR   ;IECTALK
C11E: A5B9              LDA.  .SECADR
C120: 2096FF            JSR   ;SENDSEC
C123: A490              LDY.  .ST
C125: D037              BNE   +DLIST4
C127: A006              LDY#  #6
C129: 84FB    =DLIST1   STY.  .TMP
C12B: 20A5FF            JSR   ;IECINP
C12E: A6FC              LDX.  .TMP+1
C130: 86FC              STA.  .TMP+1
C132: A490              LDY.  .ST
C134: D028              BNE   +DLIST4
C136: A4FB              LDY.  .TMP
C138: 88                DEY
C139: D0EE              BNE   +DLIST1
C13B: A4FC              LDY.  .TMP+1
C13D: 20CDBD            JSR   ;LNPRT
C140: A920              LDA#  '
C142: 20D2FF            JSR   ;BSOUT
C145: 20A5FF  =DLIST3   JSR   ;IECINP
C148: A690              LDX.  .ST
C14A: D012              BNE   +DLIST4
C14C: AA                TAX
C14D: F006              BEQ   +DLIST2
C14F: 20D2FF            JSR   ;BSOUT
C152: 4C45C             JMP   ;DLIST3
C155: A90D    =DLIST2   LDA#  #CR
C157: 20D2FF            JSR   ;BSOUT
C15A: A004              LDY#  #4
C15C: D0CB              BNE   +DLIST1
C15E: 2042F6  =DLIST4   JSR   ;CLOSEFI
C161: 4CAEA7            JMP   ;INTERP
              * TABELLE FUER 'FEHLER' FUER DEN DERROR-BEFEHL
C164:         =DERRTAB        &ERR
C164: 455252
C167: B0                      .ORTOKEN
```

10

Wie man Gerätetreiber hinzufügt

Wie man Gerätetreiber hinzufügt

Die Ein-/Ausgabetreiber des C64 Betriebssystems (OS) werden über Vektoren angesprungen. Diese Vektoren zeigen auf Betriebssystemprogramme, die die Ein-/Ausgabe bedienen. Deshalb kann der Anwender diese Vektoren ändern und sie auf seine eigenen Programme zeigen lassen.

Zuerst sollten Sie überprüfen, ob Ihr definiertes Gerät angesprochen ist. Ist das der Fall, arbeiten Sie mit Ihrem eigenen Gerätetreiber, sonst springen Sie in den Treiber des Betriebssystems (OS). Das nächste Kapitel beschreibt einen Treiber für eine Centronics-Schnittstelle.

Nachfolgend eine Beschreibung der für die Ein-/Ausgabe verwendeten Vektoren:

031A - 031B	OPEN Vektor zeigt auf das Unterprogramm, das das Gerät aus Speicherstelle $BA öffnet und initialisiert.
031C - 031D	CLOSE Vektor Schließt die Datei (Nummer im Akku) und das zu ihr gehörende Gerät.
031E - 031F	CHKIN Vektor Speichert die zur Dateinummer gehörende Gerätenummer aus dem X-Register auf Speicherstelle $99 ab.
0320 -0321	CKOUT Vektor Wie CHKIN, die Gerätenummer wird jedoch auf Speicherstelle $9A (Aktuelles Ausgabegerät) abgelegt.
0322 - 0323	CLRCH Vektor Setzt die Standard-Ein-/Ausgabegeräte und setzt die aktuellen Geräte zurück.

0324 - 0325	INPUT Vektor Holt Zeichen vom Gerät, das in Speicherstelle $99 angegeben ist. Das Zeichen muß im Akku stehen, kein anderes Register darf verändert werden.
0326 - 0327	OUTPUT Vektor Gibt ein Zeichen (im Akku) an das Gerät aus, das an Speicherstelle $9A steht. Kein anderes Register darf verändert werden.
032A - 032B	GET Vektor Holt ein Zeichen vom Eingabegerät, das in Speicherstelle $99 steht. Sind keine Daten vorhanden, wird Null zurückgegeben. Der Akku enthält das Zeichen, die anderen Register dürfen nicht verändert werden.
032C - 032D	CLALL Vektor Schließt alle Dateien. Danach wird CLRCH durchlaufen.
0330 - 0331	LOAD Vektor Lädt oder verifiziert ein Programm. (Siehe auch Kapitel 4)
0332 - 0333	SAVE Vektor Siehe Kapitel 4.

Möglicherweise werden bei Ihrer Anwendung nicht alle Vektoren gebraucht. Sehen Sie sich das im nächsten Kapitel vorgestellte Beispielprogramm an, wenn Sie Ihre eigenen Unterprogramme einbinden wollen.

Verlassen Sie alle Unterprogramme mit rückgesetzten Carry-Bit, wenn kein Fehler auftritt.

11

Eine preiswerte Centronics-Schnittstelle

Eine preiswerte Centronics-Schnittstelle

Das folgende Programm ermöglicht Ihnen den Anschluß eines Drukkers mit einer Centronics-Schnittstelle über das User Port des C64.

Das Programm schaltet die RS232-Schnittstelle ab und installiert eine Centronics-Schnittstelle als Gerät 2. Die Schnittstelle überträgt 7 Bit ASCII. Bit 7 ist immer 0. Sie können die normalen Ein-/Ausgabebefehle benutzen. Eine Sekundäradresse, die nicht Null ist, schaltet den Zeilenvorschub-Modus ein. Das heißt, daß nach jedem Wagenrücklauf (= CR) ein zusätzlicher Zeilenvorschub (= LF) gesendet wird.

Die folgenden Leitungen müssen verbunden werden:

User Port	zum	Drucker	Beschreibung
L		1	STROBE
C		2	DATA1
D		3	DATA2
E		4	DATA3
F		5	DATA4
H		6	DATA5
J		7	DATA6
K		8	DATA7
N		9	DATA8
M		11	BUSY
N		19	MASSE

Wie Sie sehen, brauchen Sie lediglich zwei Steckverbinder. Für das User Port benötigen Sie einen TRW CINCH 251-12-50-170/50-24sn-98124 Steckverbinder. Welchen Steckverbinder Sie für den Drucker brauchen, müssen Sie dem Druckerhandbuch entnehmen.

Sie aktivieren die Centronics-Schnittstelle mit SYS12*4096.

```
                *        CENTRONICSSCHNITTSTELLE        *
                *        UEBER USER PORT                *
                *        SYS12*4096                     *

                @$031A          =IOVECT
                @$DD00          =CIA2
                @$DD00          =PORTA
                @$DD01          =PORTB
                @$DD02          =DDRA
                @$DD03          =DDRB
                @$F333          =CLRCH
                @$F6ED          =STOPQ
                @$F32F          =CLALL
                @13             =CR
                @10             =LF
                @19             =TABLEN
                @$C000          =ORG
                * SETZEN VON EIGENEN VEKTOREN
C000: A213      =INSTALL LDX#   .TABLEN
C002: BD0CC0    =INSTLP  LDAX   .HANDTAB
C005: 9D1A03             STAX   .IOVECT
C008: CA                 DEX
C009: 10F7               BPL    +INSTLOOP
C00B: 60                 RTS
                * VEKTORTABELLE
C00C: 20C0      =HANDTAB        ;OPEN
C00E: 6BC0                      ;CLOSE
C010: 86C0                      ;CHKIN
C012: 9DC0                      ;CKOUT
C014: 33F3                      ;CLRCH
C016: B5C0                      ;BASIN
C018: C1C0                      ;BSOUT
C01A: EDF6                      ;STOPQ
C01C: FCC0                      ;GET
C01E: 2FF3                      ;CLALL
                * DIE OPEN-ROUTINE
C020: A6B8      =OPEN    LDX.   $B8
C022: D003               BNE    +NOTZER
                * KEINE EINGABEDATEI
C024: 4C0AF7             JMP    ;$F70A
C027: 200FF3    =NOTZER  JSR    ;$F30F
C02A: D003               BNE    +NOTFOUN
                * 'FILE OPEN'-FEHLER
C02C: 4CFEF6             JMP    ;$F6FE
```

```
C02F: A698    =NOTFOUN LDX.  $98
C031: E00A             CPX#  #10
C033: 9003             BCC   +NOTFULL
              * #FILES > 10 --> FEHLERAUSGABE: 'TOO MANY FILES'
C035: 4CFBF6           JMP   ;$F6FB
C038: E698    =NOTFULL INC.  $98
              * DATEINUMMER
C03A: A5B8             LDA.  $B8
C03C: 9D5902           STAX  ;$259
              * SEKUNDAER-ADRESSE
C03F: A5B9             LDA.  $B9
C041: 8D09C1           STA   ;LFFLG
C044: 0960             ORA#  #$60
C046: 85B9             STA.  $B9
C048: 9D6D02           STAX  ;$26D
              * GERAETENUMMER
C04B: A5BA             LDA.  $BA
C04D: 9D6302           STAX  ;$263
              * GERAET 2 ?
C050: C902             CMP#  #2
C052: F003             BEQ   +CTRXOPN
              * NEIN, SPRINGE INS OS
C054: 4C72F3           JMP   ;$F372
              * CIA FUER PARALLELBETRIEB VORBEREITEN
C057: A9FF    =CTRXOPN LDA#  $FF
C059: 8D03DD           STA   ;DDRB
C05C: AD02DD           LDA   ;DDRA
C05F: 29FB             AND#  $FB
C061: 8D02DD           STA   ;DDRA
C064: A980             LDA#  $80
C066: 8D01DD           STA   ;PORTB
C069: 18               CLC
C06A: 60               RTS
              * CLOSE
C06B: 2014F3  =CLOSE   JSR   ;$F314
C06E: F002             BEQ   +FOUND
              * WIRD KEINE DATEI GEFUNDEN --> RUECKKEHR
C070: 18               CLC
C071: 60               RTS
              * GEFUNDEN, --> PARAMETER SETZEN
C072: 201FF3  =FOUND   JSR   ;$F31F
C075: 8A               TXA
C076: 48               PHA
C077: A5BA             LDA.  $BA
```

```
                * GERAET 2 ?
C079: C902              CMP#  #2
C07B: F003              BEQ   +CTRXCLO
                * SPRINGE INS OS
C07D: 4C9DF2            JMP   ;$F29D
                * CLOSE (EIGENE ROUTINE)
C080: 68        =CTRXCLO PLA
C081: 20F2F2            JSR   ;$F2F2
                * RUECKKEHR
C084: 18                CLC
C085: 60                RTS
C086: 200FF3 =CHKIN     JSR   ;$F30F
C089: F003              BEQ   +FOUND2
                * FEHLER: FILE NOT OPEN
C08B: 4C01F7            JMP   ;$F701
C08E: 201FF3 =FOUND2    JSR   ;$F31F
C091: A5BA              LDA.  $BA
C093: C902              CMP#  #2
C095: F003              BEQ   ;CTRXCKI
C097: 4C19F2            JMP   ;$F219
C09A: 4C0AF7 =CTRXCKI   JMP   ;$F70A
C09D: 200FF3 =CKOUT     JSR   ;$F30F
C0A0: F003              BEQ   +FOUND3
                * FEHLER: DATEI NICHT GEFUNDEN
C0A2: 4C01F7            JMP   ;$F701
C0A5: 201FF3 =FOUND3    JSR   ;$F31F
C0A8: A5BA              LDA.  $BA
                * GERAET 2 ?
C0AA: C902              CMP#  #2
C0AC: F003              BEQ   +CTRXCKO
                * NEIN, INS OS
C0AE: 4C5BF2            JMP   ;$F25B
                * JA, DANN AUSGABEDATEI SETZEN
C0B1: 859A      =CTRXCKO STA. $9A
C0B3: 18                CLC
C0B4: 60                RTS
C0B5: A599      =BASIN  LDA.  $99
C0B7: C902              CMP#  #2
C0B9: F003              BEQ   +CTRXBSI
C0BB: 4C57F1            JMP   ;$F157
                * IST EINGABEGERAET 2 --> 'NOT INPUT FILE'
C0BE: 4C0AF7 =CTRXBSI   JMP   ;$F70A
C0C1: 48        =BSOUT  PHA
C0C2: A59A              LDA.  $9A
```

```
C0C4: C902              CMP#  #2
C0C6: F004              BEQ   +CTRXBSO
C0C8: 68                PLA
C0C9: 4CCAF1            JMP   ;$F1CA
              * CENTRONIX AUSGABE
C0CC: 8E08C1 =CTRXBSO STX   ;XSAVE
C0CF: 68                PLA
C0D0: 20E8C0            JSR   ;PAROUT
C0D3: C90D              CMP#  #CR
C0D5: D00C              BNE   +NOLF
              * IST SEKUNDAERADRESSE > 0, ZUSAETZLICH ZEILENVOR-
              * SCHUB (LF) AUSGEBEN
C0D7: AE09C1            LDX   ;LFFLG
C0DA: F007              BEQ   +NOLF
C0DC: A90A              LDA#  .LF
C0DE: 20E8C0            JSR   ;PAROUT
C0E1: A90D              LDA#  .CR
C0E3: AE08C1 =NOLF      LDX   ;XSAVE
C0E6: 18                CLC
C0E7: 60                RTS
              * PARALLELAUSGABE
C0E8: 48      =PAROUT   PHA
              * WARTEN, BIS DER DRUCKER BEREIT IST
C0E9: AD00DD =WAIT      LDA   ;PORTA
C0EC: 2904              AND#  #4
C0EE: D0F9              BNE   +WAIT
C0F0: 68                PLA
              * STROBE (BIT7) SETZEN
              * AUSGABE DER DATEN (BIT0-6) AUF DEN BUS
C0F1: 0980              ORA#  $80
C0F3: 8D01DD            STA   ;PORTB
              * STROBE RUECKSETZEN BIS ZUM NAECHSTEN ZU
              * SENDENDEN BYTE
C0F6: 297F              AND#  $7F
C0F8: 8D01DD            STA   ;PORTB
C0FB: 60                RTS
C0FC: A599    =GET      LDA.  $99
C0FE: C902              CMD#  #2
C100: F003              BEQ   +CTRXGET
C102: 4C3EF1            JMP   ;$F13E
              * FEHLERAUSGABE 'NOT INPUT FILE'
C105: 4C0AF7 =CTRXGET JMP   ;$F70A
              * TEMPORAERES XSAVE
C108: 00      =XSAVE          #0
```

```
            * ABLEGEN DER SEKUNDAERADRESSE FUER ZUSAETZ-
            * LICHEN ZEILENVORSCHUB
C109: 00    =LFFLG          #0

ASSEMBLIERUNG BENDET
```

12

Formatierter Ausdruck

Formatierter Ausdruck

Das nächste Programm hilft Ihnen, Ihr Programm innerhalb definierter Randmarkierungen auszudrucken. Keine Zeile geht über eine vordefinierte Zeichenlänge hinaus.

Das Programm ändert den Ausgabevektor ($0326-$0327). Mit SYS12*4096 zeigt der Vektor auf Ihr eigenes Unterprogramm. Der eingestellte Wert für den rechten Rand ist 40, aber Sie können den Wert jederzeit ändern (0-255). Ihr Programmausdruck wird übersichtlicher, aber länger, da zusätzliche CR-LF's eingefügt werden, wenn eine Zeile die maximale Länge überschreitet.

```
                 *        FORMATIERTER AUSDRUCK        *

                 @$0326          =OUTVEC
                 @13             =CR
                 @$C000          =ORG
                 * SETZT EINEN NEUEN AUSGABEVEKTOR;
                 * DER ALTE WIRD IN OLDOUT ZWISCHENGESPEICHERT
                 * AUFRUF MIT: SYS12*4096
C000: AD2603 =INSTALL LDA    ;OUTVEC
C003: 8D4CC0          STA    ;OLDOUT
C006: AD2703          LDA    ;OUTVEC+1
C009: 8D4DC0          STA    ;OLDOUT+1
C00C: A930            LDA#   .NEWOUTL
C00E: 8D2603          STA    ;OUTVEC
C011: A9C0            LDA#   .NEWOUTH
C013: 8D2703          STA    ;OUTVEC+1
                 * DIE NEUE AUSGABEROUTINE
                 @$C0            =NEWOUTH
                 @$30            =NEWOUTL
                 @$C030          =NEWOUT
C030: C90D            CMP#   .CR
C032: F008            BEQ    +DOCR
C034: CE4EC0          DEC    ;COUNT
C037: D00B            BNE    +NADDCR
C039: 2048C0          JSR    ;BSOUT2
C03C: AD4BC0 =DOCR    LDA    ;CPERL
C03F: 8D4EC0          STA    ;COUNT
```

Formatierter Ausdruck

```
C042: A90D              LDA#  .CR
C044: 2048C0 =NADDCR    JSR   ;BSOUT2
C047: 60                RTS
             * ALTE AUSGABE
C048: 6C4CC0 =BSOUT2    JMP@  ;OLDOUT
             * SPEICHERSTELLE, DIE DIE RECHTE RANDPOSITION
             * ENTHAELT (49227)
C04B: 28     =CPERL           #40
C04C: 0000   =OLDOUT          ;0
C04E: 00     =COUNT           #0
```

ASSEMBLIERUNG BEENDET

13

Ausgabe des Bildinhalts auf einen Drucker

Ausgabe des Bildinhalts auf einen Drucker

Das nächste Programm erstellt einen Ausdruck des Bildschirminhalts (Hardcopy). Es kann direkt in BASIC mit SYS12*4096 angesprungen werden. Der Drucker muß der Gerätenummer 4 zugeordnet sein. Alle grafischen Zeichen werden ausgedruckt, da das Programm den Bildschirm-Code in ASCII-Zeichen umwandelt. Invers dargestellte Zeichen werden normal dargestellt.

Es könnte eine interessante Übung für Sie sein, das Programm zu erweitern und auch invers dargestellte Zeichen auszudrucken.

```
                *       AUSDRUCK DES BILDSCHIRM-      *
                *       INHALTS                       *
                *       SYS12*4096                    *

                @$FC            =TEMP
                @$FD            =PT
                @0              =SCREENL
                @4              =SCREENH
                @$FFBA          =SETFPA
                @$FFBD          =SETFNA
                @$FFC0          =OPEN
                @$FFC3          =CLOSE
                @$FFC9          =CKOUT
                @$FFCC          =CLRCH
                @$FFD2          =BSOUT
                @$FFE1          =STOPQ
                @40             =COLUMN
                @25             =ROW
                @13             =CR
                @C000           =ORG
                * DATEI-, GERAETENUMMER UND SEKUNDAERADRESSE SETZEN
C000: A97F              LDA#  #127
C002: A204              LDX#  #4
C004: A000              LDY#  #0
C006: 20BAFF            JSR   ;SETFPA
                * KEIN DATEINAME
C009: A900              LDA#  #0
C00B: 20BDFF            JSR   ;SETFNA
```

Ausgabe des Bildinhalts auf einen Drucker

```
                * DATEI EROEFFNEN
C00E: 20C0FF            JSR   ;OPEN
C011: B04A              BCS   +QUIT
                * BILDSCHIRMANFANG
C013: A900              LDA#  .SCREENL
C015: 85FD              STA.  .PT
C017: A904              LDA#  .SCREENH
C019: 85FE              STA.  .PT+1
                * GERAETEAUSGABEADRESSE
C01B: A27F              LDX#  #127
C01D: 20C9FF            JSR   ;CKOUT
                * ANZAHL DER ZEILEN
C020: A219              LDX#  .ROW
                * SCHLEIFE FUER DEN AUSDRUCK
C022: A90D      =ROWLOOP LDA# .CR
C024: 20D2FF            JSR   ;BSOUT
C027: 20E1FF            JSR   ;STOPQ
C02A: F031              BEQ   +QUIT
C02C: A000              LDY#  #0
C02E: B1FD      =COLLOOP LDA@Y .PT
                * UMWANDELN IN BILDSCHIRMCODE
C030: 85FC              STA.  .TEMP
C032: 293F              AND#  $3F
C034: 06FC              ASL.  .TEMP
C036: 24FC              BIT.  .TEMP
C038: 1002              BPL   +NOOR
C03A: 0980              ORA#  $80
C03C: 7002      =NOOR   BVS   +NOADDOR
C03E: 0940              ORA#  $40
C040: 20D2FF =NOADDOR JSR     +BSOUT
C043: C8                INY
C044: C028              CPY#  .COLUMN
C046: D0E6              BNE   +COLLOOP
C048: 98                TYA
C049: 18                CLC
C04A: 65FD              ADC.  .PT
C04C: 85FD              STA.  .PT
C04E: 9002              BCC   +NOOR1
C050: E6FE              INC.  .PT+1
C052: CA        =NOOR1  DEX
C053: D0CD              BNE   +ROWLOOP
                * ALLE ZEILEN ABGEARBEITET?
C055: A90D              LDA#  .CR
C057: 20D2FF            JSR   ;BSOUT
```

```
C05A: 20CCFF          JSR   ;CLRCH
C05D: A97F   =QUIT    LDA#  #127
C05F: 4CC3FF          JMP   ;CLOSE

ASSEMBLIERUNG BEENDET
```

Bildschirmausdruck über die RS232-Schnittstelle

Für den Ausdruck des Bildschirminhalts auf einem Drucker mit RS232-Schnittstelle kann das nachstehende Programm benutzt werden. Es gleicht dem vorangegangenen, bis auf einige kleine Details: Die Grafikzeichen können nicht ausgedruckt werden, da der Drucker sie nicht darstellen kann.

Im Programm wurde die RS232-Schnittstelle folgendermaßen initialisiert:

300 Baud
8 Datenbits
2 Stopbits
3 Handshake-Leitungen
Half Duplex
Keine Parität

Es steht Ihnen frei, sie den Anforderungen Ihres Druckers anzupassen.

```
*        AUSDRUCK DES BILDSCHIRM-      *
*        INHALTS UEBER RS232           *
*        SYS12*4096                    *

@$FC          =TEM
@$FD          =PT
@0            =SCREENL
@4            =SCREENH
@$FFBA        =SETFPA
@$FFBD        =SETFNA
@$FFC0        =OPEN
```

Ausgabe des Bildinhalts auf einen Drucker

```
                @$FFC3          =CLOSE
                @$FFC9          =CKOUT
                @$FFCC          =CLRCH
                @$FFD2          =BSOUT
                @$FFE1          =STOPQ
                @40             =COLUMN
                @25             =ROW
                @13             =CR
                @10             =LF
                * PARAMETER FUER RS232
                @$86            =CONTROL
                @$10            =COMMAND
                @$C000          =ORG
                * DATEI-, GERAETENUMMER UND SEKUNDAERADRESSE SETZEN
C000: A97F              LDA#  #127
C002: A202              LDX#  #2
C004: A000              LDY#  #0
C006: 20BAFF            JSR   ;SETFPA
                * DATEINAME UND LAENGE (2BYTE) SETZEN
C009: A902              LDA#  #2
C00B: A200              LDX#  .FNAMEL
C00D: A0C1              LDY#  .FNAMEH
C00F: 20BDFF            JSR   ;SETFPA
                * DATEI EROEFFNEN
C012: 20C0FF            JSR   ;OPEN
                * BILDSCHIRMANFANG
C015: A900              LDA#  .SCREENL
C017: 85FD              STA.  .PT
C019: A904              LDA#  .SCREENH
C01B: 85FE              STA.  .PT+1
                * GERAETEAUSGABEADRESSE
C01D: A27F              LDX#  #127
C01F: 20C9FF            JSR   ;CKOUT
                * DRUCKSCHLEIFE
C022: A219              LDX#  .ROW
C024: A90D    =ROWLOOP  LDA#  .CR
C026: 20D2FF            JSR   ;BSOUT
C029: A90A              LDA#  .LF
C02B: 20D2FF            JSR   ;BSOUT
C02E: 20E1FF            JSR   ;STOPQ
C031: F036              BEQ   +QUIT
C033: A000              LDY#  #0
C035: B1FD    =COLLOOP  LDA@Y .PT
C037: 85FC              STA.  .TEMP
```

```
C039: 293F              AND#  $3F
C03B: 06FC              ASL.  .TEMP
C03D: 24FC              BIT.  .TEMP
C03F: 1002              BPL   +NOOR
C041: 0980              ORA#  $80
C043: 7002    =NOOR     BVS   +NOADDOR
C045: 0940              ORA#  $40
C047: 20D2FF =NOADDOR   JSR   ;BSOUT
C04A: C8                INY
C04B: C028              CPY#  .COLUMN
C04D: D0E6              BNE   +COLLOOP
C04F: 98                TYA
C050: 18                CLC
C051: 65FD              ADC.  .PT
C053: 85FD              STA.  .PT
C055: 9002              BCC   +NOOR1
C057: E6FE              INC.  .PT+1
C059: CA      =NOOR1    DEX
C05A: D0C8              BNE   +ROWLOOP
              * ALLE ZEILEN ABGEARBEITET?
C05C: A90D              LDA#  .CR
C05E: 20D2FF            JSR   ;BSOUT
C061: A90A              LDA#  .LF
C063: 20D2FF            JSR   ;BSOUT
C066: 20CCFF            JSR   ;CLRCH
C069: A97F    =QUIT     LDA#  #127
C06B: 4CC3FF            JMP   ;CLOSE
              * KONTROL- UND BEFEHLSBYTE
              @$C1            =FNAMEH
              @$0             =FNAMEL
              @$C100          =FNAME
C100: 86                      .CONTROL
C101: 10                      .COMMAND

ASSEMBLIERUNG BEENDET
```

14

Terminal

Terminal

Dieses Programm vernlaßt Ihren Commodore 64, ein Terminal zu simulieren. Die Parameter werden dabei mittels Menütechnik gesetzt. Diese Technik erspart dem Anwender das Lesen von langen Anleitungen. Ein Menü besteht aus drei unterschiedlichen Teilen:

1. Aufzählen der Wahlmöglichkeiten.

2. Wahl und Überprüfung auf Richtigkeit.

3. Ausführen des gewählten Programmteils.

Zuerst erscheint ein Übersichts-Menü. Sie wählen dann und das Programm beginnt mit der entsprechenden Abarbeitung. Sie können Menüs und Untermenüs erstellen. Sie müssen nur den Pfad wählen.

Im folgenden Programm geben Sie in Menütechnik die Parameter ein, die zum Initialisieren des Terminals notwendig sind. Haben Sie alle gewünschten Daten eingegeben, beginnt das Programm, als Terminal zu arbeiten. Sie können Ihren C64 über ein Modem oder direkt mit einen anderem Rechner verbinden. Für den Anschluß an ein Modem 300 von Hayes, das im Half-Duplex-Modus betrieben wird, sind folgende Parameter zu setzen:

300 Baud
7 Bits
1 Stopbit
3 Handshake-Leitungen
Half-Duplex
Gerade Parität

Sie können nun die Nummer des Rechners, den Sie angeschlossen haben, anwählen. Als Übung können Sie ein Unterprogramm schreiben, das Daten von und zum Rechner schickt. Verwenden Sie dabei die Funktionstasten.

In BASIC starten Sie das Terminalprogramm mit SYS12*4096. Sie können nach BASIC zurückkehren, wenn Sie gleichzeitig die Tasten RUN/STOP und RESTORE drücken.

```
                *        TERMINAL                          *
                *        SYS12*4096                        *

                @$90            =ST
                @$8E            =AUX
                @$0291          =SHFTCOM
                @$D018          =CHARGEN
                @$E716          =SCRNOUT
                @$F142          =FROMKBD
                @$FFBA          =SETFPA
                @$FFBD          =SETFNA
                @$FFC0          =OPEN
                @$FFC3          =CLOSE
                @$FFC9          =CKOUT
                @$FFCC          =CLRCH
                @$FFD2          =BSOUT
                @$FFE4          =GET
                @$FFE7          =CLALL
                @08             =BS
                @10             =LF
                @13             =CR
                @20             =DEL
                @147            =CLS
                @175            =CURS
                @157            =BACKS
                @$C000          =ORG
C000: A900              LDA#    #0
C002: 8D80C3            STA     ;CONTROL
C005: 8D81C3            STA     ;COMMAND
                * BAUDRATE
C008: 20BFC2            JSR     ;CLRSCRN
C00B: 20C4C2            JSR     ;PRINT
C00E:                           &0. 50     BAUD
C00E: 302E20
C011: 353020
C014: 202020
C017: 424155
C01A: 44
C01B: 0D                        .CR
C01C:                           &1. 75     BAUD
C01C: 312E20
C01F: 373520
C022: 202020
C025: 424155
```

```
C028: 44
C029: 0D                    .CR
C02A:                       &2. 110   BAUD
C02A: 322E20
C02D: 313130
C030: 202020
C033: 424155
C036: 44
C037: 0D                    .CR
C038:                       &3. 134.5 BAUD
C038: 332E20
C03B: 313334
C03E: 2E3520
C041: 424155
C044: 44
C045: 0D                    .CR
C046:                       &4. 150   BAUD
C046: 342E20
C049: 313530
C04C: 202020
C04F: 424155
C052: 44
C053: 0D                    .CR
C054:                       &5. 300   BAUD
C054: 352E20
C057: 333030
C05A: 202020
C05D: 424155
C060: 44
C061: 0D                    .CR
C062:                       &6. 600   BAUD
C062: 362E20
C065: 363030
C068: 202020
C06B: 424155
C06E: 44
C06F: 0D                    .CR
C070:                       &7. 1200  BAUD
C070: 372E20
C073: 313230
C076: 302020
C079: 424155
C07C: 44
C07D: 0D                    .CR
```

```
C07E:                          &8. 1800  BAUD
C07E: 382E20
C081: 313830
C084: 302020
C087: 424155
C08A: 44
C08B: 0D                       .CR
C08C:                          &9. 2400  BAUD
C08C: 392E20
C08F: 323430
C092: 302020
C095: 424155
C098: 44
C099: 8D                       .CR+128
C09A: 20E6C2          JSR      ;NINPUT
C09D: A8              TAY
C09E: AD80C3          LDA      ;CONTROL
C0A1: 19F6C2          ORAY     ;BAUDTAB
C0A4: 8D80C3          STA      ;CONTROL
            * DATENBITS
C0A7: 20BFC2          JSR      ;CLRSCRN
C0AA: 20C4C2          JSR      ;PRINT
C0AD:                          &0. 8 DATENBIT
C0AD: 302E20
C0B0: 382044
C0B3: 415445
C0B6: 4E4249
C0B9: 54
C0BA: 0D                       .CR
C0BB:                          &1. 7 DATENBIT
C0BB: 312E20
C0BE: 372044
C0C1: 415445
C0C4: 4E4249
C0C7: 54
C0C8: 0D                       .CR
C0C9:                          &2. 6 DATENBIT
C0C9: 322E20
C0CC: 362044
C0CF: 415445
C0D2: 4E4249
C0D5: 54
C0D6: 0D                       .CR
C0D7:                          &3. 5 DATENBIT
```

```
C0D7: 332E20
C0DA: 352044
C0DD: 415445
C0E0: 4E4249
C0E3: 54
C0E4: 8D                        .CR+128
C0E5: 20E6C2 =GETDAT  JSR       ;NINPUT
C0E8: C904            CMP#      #4
C0EA: B0F9            BCS       +GETDAT
C0EC: A8              TAY
C0ED: AD80C3          LDA       ;CONTROL
C0F0: 1900C3          ORAY      ;BITTAB
C0F3: 8D80C3          STA       ;CONTROL
            * STOPBITS
C0F6: 20BFC2          JSR       ;CLRSCRN
C0F9: 20C4C2          JSR       ;PRINT
C0FC:                           &0. 1 STOPBIT
C0FC: 302E20
C0FF: 312053
C102: 544F50
C105: 424954
C108: 0D                        .CR
C109:                            &1. 2 STOPBITS
C109: 312E20
C10C: 322053
C10F: 544F50
C112: 424954
C115: 53
C116: 8D                        .CR+128
C117: 20E6C2 =GETSTOP JSR       ;NINPUT
C11A: C902            CMP#      #2
C11C: B0F9            BCS       +GETSTOP
C11E: A8              TAY
C11F: AD80C3          LDA       ;CONTROL
C122: 1904C3          ORAY      ;STOPTAB
C125: 8D80C3          STA       ;CONTROL
            * HANDSHAKE-MODUS
C128: 20BFC2          JSR       ;CLRSCRN
C12B: 20C4C2          JSR       ;PRINT
C12E:                           &0. 0-3 LINE HANDSHAKE
C12E: 302E20
C131: 302D33
C134: 204C49
C137: 4E4520
```

```
C13A: 48414E
C13D: 445348
C140: 414B45
C143: 0D                    .CR
C144:                       &1.   X LINE HANDSHAKE
C144: 312E20
C147: 202058
C14A: 204C49
C14D: 4E4520
C150: 48414E
C153: 445348
C156: 414B45
C159: 8D                    .CR+128
C15A: 20E6C2 =GETHAND JSR   ;NINPUT
C15D: C902            CMP#  #2
C15F: B0F9            BCS   +GETHAND
C161: A8              TAY
C162: AD81C3          LDA   ;COMMAND
C165: 1906C3          ORAY  ;HANDTAB
C168: 8D81C3          STA   ;COMMAND
            * DUPLEX
C16B: 20BFC2          JSR   ;CLRSCRN
C16E: 20C4C2          JSR   ;PRINT
C171:                       &0. HALF DUPLEX
C171: 302E20
C174: 48414C
C177: 462044
C17A: 55504C
C17D: 4558
C17F: 0D                    .CR
C180:                       &1. FULL DUPLEX
C180: 312E20
C183: 46554C
C186: 4C2044
C189: 55504C
C18C: 4558
C18E: 8D                    .CR+128
C18F: 20E6C2 =GETDUPL JSR   ;NINPUT
C192: C902            CMP#  #2
C194: B0F9            BCS   +GETDUPL
C196: A8              TAY
C197: AD81C3          LDA   ;COMMAND
C19A: 1908C3          ORAY  ;DUPLTAB
C19D: 8D81C3          STA   ;COMMAND
```

```
                * PARITAETSMODUS
C1A0: 20BFC2            JSR   ;CLRSCRN
C1A3: 20C4C2            JSR   ;PRINT
C1A6:                         &0. KEINE PARITAET
C1A6: 302E20
C1A9: 4B4549
C1AC: 4E4520
C1AF: 504152
C1B2: 495441
C1B5: 4554
C1B7: 0D                      .CR
C1B8:                         &1. PAR. UNGER.
C1B8: 312E20
C1BB: 504152
C1BE: 2E2055
C1C1: 4E4745
C1C4: 532E
C1C6: 0D                      .CR
C1C7:                         &2. PAR. GERADE
C1C7: 322E20
C1CA: 504152
C1CD: 2E2047
C1D0: 455241
C1D3: 44445
C1D5: 0D                      .CR
C1D6:                         &3. MARK UEBERTRAGNG
C1D6: 332E20
C1D9: 4D4152
C1DC: 4B2055
C1DF: 454245
C1E2: 525452
C1E5: 41474E
C1E8: 47
C1E9: 0D                      .CR
C1EA:                         &4. SPACE UEBERTRAGNG
C1EA: 342E20
C1ED: 535041
C1F0: 434520
C1F3: 554542
C1F6: 455254
C1F9: 524147
C1FC: 4E47
C1FE: 8D                      .CR+128
```

```
C1FF: 20E6C2 =GETPARI JSR   ;NINPUT
C202: C905            CMP#  #5
C204: B0F9            BCS   +GETPARI
C206: A8              TAY
C207: AD81C3          LDA   ;COMMAND
C20A: 190AC3          ORAY  ;PARTAB
C20D: 8D81C3          STA   ;COMMAND
            * KLEINBUCHSTABEN WERDEN UNTERDRUECKT <SHFT> <C=>
C210: A917            LDA#  #23
C212: 8D18D0          STA   ;CHARGEN
C215: A9FF            LDA#  #255
C217: 8D9102          STA   ;SHFTCOM
            * MELDUNG 'TERMINAL'
C21A: 20BFC2          JSR   ;CLRSCRN
C21D: 20C4C2          JSR   ;PRINT
C220:                       &TERMINAL
C220: 544552
C223: 4D494E
C226: 414C
C228: 0D                    .CR
C229: 8D                    .CR+128
            * DATEI NUMMER 3 SCHLIESSEN
C22A: A903            LDA#  #3
C22C: 20C3FF          JSR   ;CLOSE
            * GERAET, SEKUNDAERADRESSE DATEINUMMER SETZEN
C22F: A202            LDX#  #2
C231: A000            LDY#  #0
C233: A903            LDA#  #3
C235: 20BAFF          JSR   ;SETFPA
            * DATEINAME UND LAENGE SETZEN
C238: A280            LDX#  .CONTROLL
C23A: A0C3            LDY#  .CONTROLH
C23C: A902            LDA#  #2
C23E: 20BDFF          JSR   ;SETFNA
            * OPEN
C241: 20C0FF          JSR   ;OPEN
            * EIN- UND AUSGABEGERAET SETZEN
C244: A203            LDX#  #3
C246: 20C6FF          JSR   ;CHKIN
C249: A203            LDX#  #3
C24B: 20C9FF          JSR   ;CKOUT
            * EIGENER CURSOR
C24E: A9AF            LDA#  .CURS
C250: 2016E7          JSR   ;SCRNOUT
```

```
C253: A99D            LDA#  .BACKS
C255: 2016E7          JSR   ;SCRNOUT
            * TERMIANLSCHLEIFE, BEI FEHLERSTATUS --> STOP
C258: A690  =TERMLP   LDX.  .ST
C25A: D048            BNE   +TERMERR
C25C: 2042F1          JSR   ;FROMKBD
C25F: F012            BEQ   +NOKEY
            * TASTE GEDRUECKT? - JA!
C261: C914            CMP#  .DEL
C263: D002            BNE   +NODEL
            * 'DELETE-TASTE' IN STANDARD ASCII-CODE BS ($08)
            * UMWANDELN
C265: A908            LDA#  .BS
C267: C90D  =NODEL    CMP#  .CR
C269: D005            BNE   +NOCR
C26B: 20D2FF          JSR   ;BSOUT
            * BEI CR NOCH LF HINZUFUEGEN
C26E: A90A            LDA#  .LF
C270: 20D2FF          JSR   ;BSOUT
            * EINGABE UEBER RS232
C273: 20E4FF =NOKEY   JSR   ;GET
C276: 20A8C2          JSR   ;TRANSL
C279: F0DD            BEQ   +TERMLP
            * DRUCKBARES ZEICHEN? - JA!
C27B: C90D            CMP#  .CR
C27D: D007            BNE   +TOSCRN
            * BEI CR CURSOR LOESCHEN
C27F: A920            LDA#  '
C281: 2016E7          JSR   ;SCRNOUT
C284: A90D            LDA#  .CR
            * ZEICHEN IN COMMODORE ASCII UMWANDELN
C286: 48    =TOSCRN   PHA
C287: C941            CMP#  'A
C289: 9008            BCC   +TOSCRN2
C28B: C95B            CMP#  $5B
C28D: B004            BCS   +TOSCRN2
C28F: 68              PLA
C290: 4920            EOR#  $20
C292: 48              PHA
C293: 68    =TOSCRN2  PLA
C294: 2016E7          JSR   ;SCRNOUT
            * CURSOR AN DIE NAECHSTE STELLE SETZEN
C297: A9AF            LDA#  .CURS
C299: 2016E7          JSR   ;SCRNOUT
```

```
C29C: A99D            LDA#  .BACKS
C29E: 2016E7          JSR   ;SCRNOUT
C2A1: 4C58C2          JMP   ;TERMLP
C2A4: 20E7FF =TERMERR JSR   ;CLALL
C2A7: 60              RTS
            * WANDELT ASCII-CODE IN COMMODORE-CODE UM
            * NICHT DIE 'NICHT DARSTELLBAREN'
C2A8: A014   =TRANSL  LDY#  .DEL
C2AA: 297F            AND#  $7F
C2AC: C908            CMP#  .BS
C2AE: F00D            BEQ   +TRNSRTS
C2B0: A00D            LDY#  .CR
C2B2: C90D            CMP#  .CR
C2B4: F007            BEQ   +TRNSRTS
C2B6: A000            LDY#  #0
C2B8: C920            CMP#  '
C2BA: 9001            BCC   +TRNSRTS
C2BC: A8              TAY
C2BD: 98     =TRNSRTS TYA
C2BE: 60              RTS
            * BILDSCHIRM LOESCHEN
C2BF: A993   =CLRSCRN LDA#  .CLS
C2C1: 4CD2FF          JMP   ;BSOUT
            * AUSGABE VON ZEICHENKETTEN (KAPITEL 2)
C2C4: 68     =PRINT   PLA
C2C5: 858E            STA.  .AUX
C2C7: 68              PLA
C2C8: 858F            STA.  .AUX+1
C2CA: A200            LDX#  #0
C2CC: E68F   PRINT1   INC.  .AUX
C2CE: D002            BNE   +PRINT2
C2D0: A18E            INC.  .AUX+1
C2D2: A18E   =PRINT2  LDA@X .AUX
C2D4: 297F            AND#  $7F
C2D6: 20D2FF          JSR   ;BSOUT
C2D9: A200            LDX#  #0
C2DB: A18E            LDA@X .AUX
C2DD: 10ED            BPL   +PRINT1
C2DF: A58F            LDA.  .AUX+1
C2E1: 48              PHA
C2E2: A58E            LDA.  .AUX
C2E4: 48              PHA
C2E5: 60              RTS
```

```
                * ZEICHEN VON DER TASTATUR
C2E6: 20E4FF =NINPUT  JSR   ;GET
C2E9: F0FB            BEQ   ;NINPUT
C2EB: C930            CMP#  '0
C2ED: 90F7            BCC   +NINPUT
C2EF: C93A            CMP#  ':
C2F1: B0F3            BCS   +NINPUT
C2F3: 290F            AND#  $0F
C2F5: 60              RTS
                * BAUDRATE
C2F6: 01        =BAUDTAB      #1
C2F7: 02                      #2
C2F8: 03                      #3
C2F9: 04                      #4
C2FA: 05                      #5
C2FB: 06                      #6
C2FC: 07                      #7
C2FD: 08                      #8
C2FE: 09                      #9
C2FF: 0A                      #10
                * DATENBITS
C300: 00        =BITTAB       $0
C301: 20                      $20
C302: 40                      $40
C303: 60                      $60
                * STOPBITS
C304: 00        =STOPTAB      $0
C305: 80                      $80
                * HANDSHAKE
C306: 00        =HANDTAB      #0
C307: 01                      #1
                * DUPLEX
C308: 00        =DUPLTAB      $0
C309: 10                      $10
                * PARITAET
C30A: 00        =PARTAB       $0
C30B: 20                      $20
C30C: 60                         $60
C30D: A0                      $A0
C30E: E0                      $E0
```

Terminal

```
            * DER 'DATEINAME'
            @$C3            =CONTROLH
            @$80            =CONTROLL
            @$C380          =ORG2
C380: 00    =CONTROL        #0
C381: 00    =COMMAND        #0

ASSEMBLIERUNG BEENDET
```

15

Wie man den C-64 mit einem ATARI verbindet

Wie man den C-64 mit einem Atari verbindet

Wir haben einen ATARI 800 über die eingebaute RS232-Schnittstelle mit Erfolg an den Commodore 64 angeschlossen. In unserem Fall diente der ATARI als Sende-, der C64 als Empfangsgerät. Das folgende Programm enthält nur Beispiele. Sie müssen Ihre eigenen PUT- und GET-Unterprogramme noch anfügen. Starten Sie zuerst das Empfangsgerät (C64), dann das Sendegerät (ATARI). Der C64 wartet bis ein Datenbyte kommt, das ungleich Null ist. Nach dem ersten Zeichen wartet der C64, bis neue Daten von der RS232-Schnittstelle empfangen wurden, dann verzweigt das Programm in ein vom Anwender erstelltes PUT-Programm.

Diesmal müssen nur zwei Leitungen verbunden werden. Sie dürfen das Signal nicht invertieren, da das ATARI-Programm dies per Software erledigt. Verbinden Sie die Rechner folgendermaßen:

ATARI
GAME Port 3
Pin 1 : Sendedaten
Pin 2 : Masse

COMMODORE 64
User Port
Pin B & C : Empfangsdaten
Pin N : Masse

Verbinden Sie Pin 1 mit Pin B & C und Pin 2 mit Pin N.

Meistens wird das erste Datenbyte zerstört. Sie senden deshalb besser erst ein Byte, das nicht verwendet werden muß. Die Programme senden und empfangen mit 300 Baud. Das heißt, daß die Übertragung großer Datenmengen sehr lange dauern wird. Sie können die Geschwindigkeit vergrößern, wenn Sie die Baudrate erhöhen.

```
*        EMPFANGEN VOM ATARI 800        *

@$29B          =RECPOIN
@$FFBA         =SETFPA
@$FFBD         =SETFNA
```

```
                @$FFC0          =OPEN
                @$FFC3          =CLOSE
                @$FFC6          =CHLIN
                @$FFCC          =CLRCH
                @$FFE4          =GET
                * ANWENDERDEFINIERT
                @$B000          =PUT
                @$C000          =ORG
C000: A902              LDA#    #2
C002: A202              LDX#    #2
C004: A000              LDY#    #0
C006: 20BAFF            JSR     ;SETFPA
C009: A902              LDA#    #2
C00B: A200              LDX#    .NAMEL
C00D: A0C1              LDY#    .NAMEH
C00F: 20BDFF            JSR     ;SETFNA
C012: 20C0FF            JSR     ;OPEN
C015: A202              LDX#    #2
C017: 20C6FF            JSR     ;CHKIN
C01A: 20E4FF =WAIT      JSR     ;GET
C01D: F0FB              BEQ     +WAIT
C01F: AD9B02            LDA     ;RECPOIN
C022: 8D02C1            STA     ;OLDPOIN
C025: 4C2DC0            JMP     ;WAIT2
C028: 2000B0 =LOOP      JSR     ;PUT
                * CARRY GESETZT --> BUFFER VOLL
C02B: B011              BCS     +READY
                * AUF ZEICHEN WARTEN
C02D: AD9B02 =WAIT2     LDA     ;RECPOIN
C030: CD02C1            CMP     ;OLDPOIN
C033: F0F8              BEQ     +WAIT2
C035: 8D02C1            STA     ;OLDPOIN
C038: 20E4FF            JSR     ;GET
C03B: 4C28C0            JMP     ;LOOP
C03E: A902   =READY     LDA#    #2
C040: 20C3FF            JSR     ;CLOSE
C043: 20CCFF            JSR     ;CLRCH
C046: 00                BRK
                @$C1            =NAMEH
                @$0             =NAMEL
                @C100           =NAME
C100: 86                        $86
C101: 10                        $10
C102: 00     =OLDPOIN           #0
```

```
                *        ATARIPROGRAMM: SENDEN         *

                @$1F            =COUNT
                @$3303          =GET
                @$D303          =PACTL
                @$D301          =PORTA
                @$D40E          =NMIEN
                @$D400          =DMACTL
                * K UND L FUER 300 BAUD
                @150            =K
                @6              =L
                @$AB00          =ORG
AB00: A930              LDA#  $30
AB02: 8D03D3            STA   ;PACTL
AB05: A901              LDA#  #1
AB07: 8D01D3            STA   ;PORTA
AB0A: A934              LDA#  $34
AB0C: 8D03D3            STA   ;PACTL
AB0F: A901              LDA#  #1
AB11: 8D01D3            STA   ;PORTA
AB14: 2067AB            JSR   ;BITWAIT
AB17: 2067AB            JSR   ;BITWAIT
AB1A: 200333 =LOOP      JSR   ;GET
AB1D: 48                PHA
AB1E: 2027AB            JSR   ;SEROUT
AB21: 68                PLA
                * BEI CTRL-Z ANHALTEN
AB22: C91A              CMP#  #26
AB24: D0F4              BNE   +LOOP
AB26: 00                BRK
                * SERIELLE AUSGABE
AB27: 8D72AB =SEROUT    STA   ;BUFFER
                * KEINE INTERRUPTS ZULASSEN
AB2A: 78                SEI
AB2B: A900              LDA#  #0
AB2D: 8D0ED4            STA   ;NMIEN
AB30: 8D00D4            STA   ;DMACTL
                * STARTBIT SENDEN
AB33: A900              LDA#  #0
AB35: 8D01D3            STA   ;PORTA
AB38: 2067AB            JSR   ;BITWAIT
                * BYTE SENDEN
AB3B: A008              LDY#  #8
AB3D: 841F              STY.  .COUNT
```

```
AB3F: AD72AB =SENDBYT LDA    ;BUFFER
AB42: 8D01D3          STA    ;PORTA
AB45: 6A              RORA
AB46: 8D72AB          STA    ;BUFFER
AB49: 2067AB          JSR    ;BITWAIT
AB4C: C61F            DEC    .COUNT
AB4E: D0EF            BNE    +SENDBYT
            * 2 STOPBITS SENDEN
AB50: A901            LDA#   #1
AB52: 8D01D3          STA    ;PORTA
AB55: 2067AB          JSR    ;BITWAIT
AB58: 2067AB          JSR    ;BITWAIT
            * INTERRUPTS ZULASSEN
AB5B: A922            LDA#   $22
AB5D: 8D00D4          STA    ;DMACTL
AB60: A9FF            LDA#   $FF
AB62: 8D0ED4          STA    ;NMIEN
AB65: 58              CLI
AB66: 60              RTS
            * UNTERPROGRAMM FUER GENAUE BAUDRATE
AB67: A296  =BITWAIT LDX#   .K
AB69: A006  =LOOPK   LDY#   .L
AB6B: 88    =LOOPL   DEY
AB6C: D0FD            BNE    +LOOPL
AB6E: CA              DEX
AB6F: D0F8            BNE    +LOOPK
AB71: 60              RTS
            * UNTERPROGRAMM FUER DEN RS232 TREIBER
            * 1 BYTE BUFFER
AB72: 00    =BUFFER          #0

ASSEMBLIERUNG BEENDET
```

16

Die RESTORE-Taste

Die RESTORE-Taste

Auf der rechten Seite Ihrer C64-Tastatur befindet sich eine Taste mit der Aufschrift 'RESTORE'. Welche Funktion hat diese Taste?

Vielleicht haben Sie sie schon in BASIC zusammen mit der RUN/STOP-Taste für einen BASIC-Warmstart verwendet. Was passiert, wenn man die RESTORE-Taste drückt? Es wird ein NMI-Interrupt ausgelöst und der Programmzähler des Bausteins 6510 wird mit der Adresse, die an den Speicherstellen $FFFA und $FFFB abgelegt ist, geladen. Diese Speicherstellen liegen im ROM und die abgespeicherte Adresse lautet $FE43. Sehen wir uns den sogenannten NMI-Treiber ab Adresse $FE43 an.

```
FE43  78      SEI
FE44  6C1803  JMP@  ;$0318       JMP $FE47
FE47  48      PHA
FE48  8A      TXA
FE49  48      PHA
FE4A  98      TYA
FE4B  48      PHA
FE4C  A97F    LDA#  $7A
FE4E  8D0DDD  STA   ;$DD0D
FE51  AC0DDD  LDY   ;$DD0D
FE54  301C    BMI   +$FE72       RS232?
FE56  2002FD  JSR   ;$FD02       ROM IN $8000?
FE59  D003    BNE   +$FE5E       NEIN!
FE5B  6C0280  JMP@  ;$8002       JA! INS ROM SPRINGEN
FE5E  20BCF6  JSR   ;$F6BC       FLAG FUER STOP SETZEN
FE61  20E1FF  JSR   ;$FFE1       STOP TASTE?
FE64  D00C    BNE   +$FE72       NEIN! DANN ABFRAGE RS232
FE66  2015FD  JSR   ;$FD15       INITIALISIERUNG DER IO VEKTOREN
FE69  20A3FD  JSR   ;$FDA3       IO INITIALISIERUNG
FE6C  2018E5  JSR   ;$E518       IO INIT. BILDSCHIRM LOESCHEN
FE6F  6C02A0  JMP@  ;$A002       BASIC WARMSTART

FE72 - FEBB   RS232 TREIBER

FEBC  68      PLA                REGISTERINHALTE WIEDERHOLEN
FEBD  A8      TAY
FEBE  68      PLA
```

```
FEBF  AA     TAX
FEC0  68     PLA
FEC1  40     RTI                    ZURUECK INS PROGRAMM
```

Sie können also den Vektor auf $0318-$0319 ändern und ihn auf Ihren eigenen NMI-Treiber zeigen lassen. Sie müssen vorher die Register sichern und prüfen, ob die RS232-Schnittstelle in Ihrem eigenen Treiber aktiv ist. Ist das der Fall, müssen Sie nach $FE72 springen. Sonst können Sie Ihr Programm neu starten.

Das folgende Programm ist ein Beispiel.

```
             *        RESET UEBER RESTORE          *
             *        VERWENDUNG VON NMI           *

             @$318          =NMIVECT
             @$FE72         =OTHMNI
             @$F6BC         =STSTOPB
             @$FFE1         =STOPQ
             @$FD15         =SETVECT
             @$FDA3         =INITIO1
             @$E518         =INITIO2
             @$810          =USER
             @$C0           =NMIHNDH
             @$B            =NMIHNDL
             @$C000         =ORG
             * DEN NMIVEKTOR IN TREIBER SETZEN
C000: A90B   =INITNMI LDA#  .NMIHNDL
C002: 8D1803          STA   ;NMIVECT
C005: A9C0            LDA#  .NMIHNDH
C007: 8D1903          STA   ;NMIVECT+1
C00A: 00              BRK
             * NMI TREIBER
C00B: 48     =NMIHND  PHA
C00C: 8A              TXA
C00D: 48              PHA
C00E: 98              TYA
C00F: 48              PHA
             * RS232 AKTIV?
C010: A97F            LDA#  $7F
```

```
C012: 8D0DDD          STA   ;$DD0D
C015: AC0DDD          LDY   ;$DD0D
C018: 301E            BMI   +RS232
            * NEIN! STOPKEY?
C01A: 20BCF6          JSR   ;SETSTOP
C01D: 20E1FF          JSR   ;STOPQ
C020: D016            BNE   +RS232
            * JA! INITIALISIEREN
C022: 2015FD          JSR   ;SETVECT
C025: 20A3FD          JSR   ;INITIO1
C028: 2018E5          JSR   ;INITIO2
            * VECTOR ZURUECKLADEN
C02B: A90B            LDA#  .NMIHNDL
C02D: 8D1803          STA   ;NMIVECT
C030: A9C0            LDA#  .NMIHNDH
C032: 8D1903          STA   ;NMIVECT+1
            * SPRINGE IN WARMSTART
C035: 4C1008          JMP   ;USER
            * ODER IN RS232 TREIBER
C038: 4C72FE =RS232   JMP   ;OTHNMI

ASSEMBLIERUNG BEENDET
```

17

Schnelle Ausgabe von Sedezimalzahlen

Schnelle Ausgabe von Sedezimalzahlen

Das nächste Programm stellt ein schnelles und kompaktes Hexdump-Programm dar. Sie müssen Anfangs- und Endadresse eines Speicherblocks angeben, den Sie auf einen Drucker ausgeben wollen. Das Programm arbeitet mit einem Drucker mit RS232-Schnittstelle. Anfangs- und Endadresse müssen in die bezeichneten Speicherstellen eingegeben werden.

```
                *        SCHNELLE AUSGABE VON SEDEZI-  *
                *        MALZAHLEN                     *

                @$80            =FROM
                @$82            =TO
                @$29D           =RS232OT
                @$29E           =RS232AC
                @$FFBA          =SETFPA
                @$FFBD          =SETFNA
                @$FFC0          =OPEN
                @$FFC3          =CLOSE
                @$FFC9          =CKOUT
                @$FFE1          =STOPQ
                @$FFE7          =CLALL
                @$FFD2          =BSOUT
                @13             =CR
                @10             =LF
                * 16 BYTE PRO ZEILE
                @16             =MAX
                @$C000          =ORG
                * DATEINUMMER 2 SCHLIESSEN
C000: A902              LDA#  #2
C002: 20C3FF            JSR   ;CLOSE
                * DATEINUMMER, GERAET UND SEKUNDAERADRESSE SETZEN
C005: A902              LDA#  #2
C007: A202              LDX#  #2
C009: A0C0              LDY#  #0
C00B: 20BAFF            JSR   ;SETFPA
                * DATEINAME UND LAENGE* SETZEN
C00E: A902              LDA#  #2
C010: A200              LDX#  .FNAMEL
```

Schnelle Ausgabe von Sedezimalzahlen

```
C012: A0C1              LDY#  .FNAMEH
C014: 20BDFF            JSR   ;SETFNA
            * DATEI EROEFFNEN
C017: 20C0FF            JSR   ;OPEN
            * AUSGABEGERAET SETZEN
C01A: A202              LDX#  #2
C01C: 20C9FF            JSR   ;CKOUT
C01F: A580    =LINELP   LDA.  .FROM
C021: C582              CMP.  .TO
C023: A581              LDA.  .FROM+1
C025: E583              SBC.  .TO+1
C027: B038              BCS   +READY
C029: A90D              LDA#  .CR
C02B: 20D2FF            JSR   ;BSOUT
C02E: A90A              LDA#  .LF
C030: 20D2FF            JSR   ;BSOUT
C033: A581              LDA.  .FROM+1
C035: 206DC0            JSR   ;PRBYTE
C038: A580              LDA.  .FROM
C03A: 206DC0            JSR   ;PRBYTE
C03D: A920              LDA#  '
C03F: 20D2FF            JSR   ;BSOUT
C042: 20E1FF            JSR   ;STOPQ
C045: F01A              BEQ   +READY
C047: A000              LDY#  #0
C049: B180    =PRLOOP   LDA@Y .FROM
C04B: 206DC0            JSR   ;PRBYTE
C04E: C8                INY
C04F: C010              CPY#  .MAX
C051: 90F6              BCC   +PRLOOP
            * ZEILENAUSGABE (MAX BYTE)
C053: 98                TYA
C054: 18                CLC
C055: 6580              ADC.  .FROM
C057: 8580              STA.  .FROM
C059: 90C4              BCC   +LINELP
C05B: E681              INC.  .FROM+1
C05D: F002              BEQ   +READY
C05F: B0BE              BCS   ;LINELP
            * DRUCKER MUSS FERTIG SEIN
C061: AD9D02  =READY    LDA   ;RS232OT
C064: CD9E02            CMP   ;RS232AC
C067: D0F8              BNE   +READY
            * CLOSE ALL
```

```
C069: 20E7FF          JSR   ;CLALL
C06C: 00              BRK
            * AKKU-INHALT ALS HEX-BYTE AUSDRUCKEN
C06D: 48    =PRBYTE   PHA
C06E: 4A              LSRA
C06F: 4A              LSRA
C070: 4A              LSRA
C071: 4A              LSRA
C072: 2078C0          JSR   ;HEXOUT
C075: 68              PLA
C076: 290F            AND#  #15
C078: C90A  =HEXOUT   CMP#  #10
C07A: B004            BCS   +ALFA
C07C: 0930            ORA#  '0
C07E: D002            BNE   +HXOUT
C080: 6936  =ALFA     ADC#  $36
C082: 4CD2FF =HXOUT   JMP   ;BSOUT
            * RS232 KONTROLL- UND
            * BEFEHLSBYTE
            @$C1              =FNAMEH
            @$0               =FNAMEL
            @$C100            =FNAME
C100: 86                      $86
C101: 10                      $10

ASSEMBLIERUNG BEENDET
```

18

Einige Tips

Einige Tips

Wie man A/D-Wandler verwendet

Der C-64 besitzt zwei A/D-Wandler. Sie können damit ein Paar Paddles anschließen. Paddles sind Potentiometer. Das heißt, die A/D-Wandler des C-64 stellen eine Art Meßgerät dar.

Der A/D-Wandler arbeitet folgendermaßen: Ein Kondensator wird auf 5 Volt aufgeladen. Register 25 (oder 26) des Bausteins SID ($D400) enthält einen Wert, der der benötigten Zeit entspricht. Dieser Vorgang wiederholt sich laufend. Damit der ganze Bereich von 8 Bit (0 - 255) ausgenutzt wird, muß der Wertebereich des Widerstands von 0 bis 500K Ohm gehen. Beim laufenden Abtasten der Werte der beiden A/D-Wandler muß eine Zeitschleife durchlaufen werden, da die Werte während des Ladens der Kondensatoren instabil sind. Da zwei Spielports vorhanden sind, können zwei Paar, also insgesammt vier Paddles, Paddles werden. Weil es aber nur zwei A/D-Wandler gibt, muß man nach einer Lösung suchen, die Paddles zu bestimmen. Für die Paddles an Spielport 1 muß $80 in die Speicherstelle abgelegt werden. Für die Paddles an Spielport 2 muß $40 abgelegt werden. Beim Arbeiten mit den Paddles muß die Tastatur abgeschaltet sein. Das wird erreicht, indem man den Interrupt (SEI) sperrt und $C0 in die Speicherstelle $DC02 ablegt, bevor man den Paddle-Wert liest. Danach muß die Tastatur wieder freigegeben werden, indem das Interruptflag (CLI) rücksetzt und $FF in $dc02 ablegt.

Natürlich sind das keine sehr genauen A/D-Wandler, aber sie können für kleine Meßaufgaben und zum Anschluß von Joysticks und Paddles verwendet werden.

ROM ins RAM kopieren

Der C-64 hat 64k Byte RAM zur Verfügung. An einigen Stellen ist ROM überlagert. Lesen Sie eine Speicherstelle, erhalten Sie den im ROM gespeicherten Wert und nicht den des RAM's. Schreiben Sie etwas in ein vom ROM überlagertes RAM, wird ins RAM geschrieben, das unterhalb des ROM liegt. Lesen Sie es zurück, erhalten Sie

den ROM-Inhalt. Schalten Sie das ROM ab, können Sie die RAM-Werte lesen. In einer RAM-Kopie des ROM-Inhalts kann das Betriebssystem geändert werden.

```
                *        KOPIERPROGRAMM                *
                *        ROM INS RAM                   *

                @2              =PT
                @4              =MAX
                @$C000          =ORG
C000: A502              LDA.   .PT
C002: C504              CMP.   .MAX
C004: A503              LDA.   .PT+1
C006: E505              SBC.   .MAX+1
C008: B00F              BCS    +READY
C00A: A200              LDX#   #2
C00C: A102      =COPY   LDA@X  .PT
C00E: 8102              STA@X  .PT
C010: E602              INC.   .PT
C012: D002              BNE    +JMPCOPY
C014: E603              INC.   .PT+1
C016: 4C0CC0    =JMPCOPY JMP   ;COPY
C019: 00        =READY  BRK

ASSEMBLIERUNG BEENDET
```

Groß- und Kleinschreibung auf dem 2022

Sie müssen für die Anwahl des Druckers für Groß- oder Kleinschreibung besondere Zeichen (Cursor nach oben und Cursor nach unten) senden. Möglicherweise wissen Sie aber nicht, daß es für den 2022 einen besonderen Modus gibt, der nicht im Handbuch aufgeführt ist, der es ermöglicht, Groß- und Kleinbuchstaben gemischt auszudrucken.

Zuerst müssen Sie den Drucker mit der Sekundäradresse 7 öffnen, ihn sofort wieder schließen und ihn erneut mit der Sekundäradresse 0 öffnen. Jetzt werden alle Zeichen auf dem 2022 ausgedruckt wie sie ankommen. Ausgenommen sind die Zeichen [ ] _ ↑ \.

Anhang

Editor und Assembler

Der Editor

Der Editor ist ein wichtiges Hilfsmittel zur Eingabe eines Textes. Viele Editorprogramme sind sehr komfortabel: automatischer Randausgleich, Hilfe beim Trennen von Wörtern und vieles mehr werden angeboten. In unserem Fall dient der Editor nur dazu, den Quelltext in einem für unseren Assembler verständlichen Format auf die Diskette zu bringen.

Dieser Editor ist eine 'Minimalversion' und besitzt nur die notwendigsten Grundfunktionen. Dennoch wird der Benutzer mit Hilfe der Menütechnik und mit entsprechenden Anweisungen geführt. Dadurch erübrigt sich eine detaillierte Diskussion der Befehle.

Beachten Sie bitte, daß Sie die Dateinamen, nach denen gefragt wird, so wählen, daß sie nicht mit bereits vorhandenen übereinstimmen. Sie verlieren sonst den ursprünglichen Dateiinhalt. Das gilt natürlich nicht für die Ausgangsdatei beim Ändern bzw. beim Ergänzen des Quelltextes.

```
100 DIM T$(100)
110 PRINTCHR$(147):PRINT
120 PRINT"  EDITOR VERSION 1.0":PRINT:PRINT
130 PRINT"   E - EINGABE AENDERN"
140 PRINT:PRINT"   N - NEUEINGABE":PRINT:PRINT:PRINT
150 PRINT"  BITTE GEBEN SIE EINEN BEFEHL EIN"
160 GET A$:IFA$=""THEN160
170 IFA$="E"THEN6000
180 IFA$="N"THEN3000
190 PRINTCHR$(147):PRINT:PRINT:PRINT:PRINT"          FEHLEINGABE"
200 FORK=0TO500:NEXTK:GOTO110
3000 REM NEUEINGABE
3010 PRINTCHR$(147)
3020 INPUT" DATEINAME";F$
3030 OPEN5,8,5,"@0:"+F$+",S,W"
3040 FORI=0TO99
3050 PRINTCHR$(147);"   NEUEINGABE":PRINT:PRINT
3060 INPUTT$(I)
3070 IFT$(I)=""THEN5000
3080 NEXT I
3090 PRINTCHR$(147):PRINT:PRINT" BITTE WARTEN"
```

```
3100 FORJ=0TO99
3105 IFT$(J)=""THEN5050
3110 PRINT#5,T$(J)
3120 NEXTJ
3125 GOSUB10000
3130 GOTO3040
5000 PRINTCHR$(147):PRINT:PRINT:INPUT" ENDE DER EINGABE (J/N)";A$
5010 IFA$="N"THENTHEN3080
5020 FORJ=0TOI-1
5025 IFT$(J)=""THEN5050
5030 PRINT#5,T$(J)
5040 NEXTJ
5050 CLOSE5:PRINTCHR$(147):END
6000 REM ZEICHENKETTEN AENDERN
6010 PRINTCHR$(147):PRINT:PRINT:PRINT
6020 PRINT" NAME DER ZU AENDERNDEN DATEI";:INPUTF1$:PRINT:PRINT:P
RINT
6030 PRINT" NAME DER GEAENDERTEN DATEI";:INPUTF2$
6040 PRINTCHR$(147);" ZEICHENKETTEN AENDERN":PRINT:PRINT:PRINT
6050 OPEN5,8,5,"@0:"+F1$+",S,R":OPEN2,8,2,"@0:"+F2$+",S,W"
6060 FORI=0TO99
6070 INPUT#5;T$(I)
6075 IFST<>0THENF=1:GOTO6090
6080 NEXTI
6090 Z=Z+1
6100 PRINT" SOLL IN DEN ";Z;". HUNDERT ZEICHENKETTEN GEAENDERT WE
RDEN (J/N)";
6110 GETA$:IFA$=""THEN6110
6120 IFA$="J"THEN8000
6130 FORI=0TO99
6135 IFT$(I)=""THEN6151
6140 PRINT#2,T$(I)
6150 NEXTI
6151 GOSUB10000:IFF=1THEN9000
6152 FORI=0TO99
6154 INPUT#5,T$(I)
6156 IFST<>0THENF=1:PRINTCHR$(147):GOTO6090
6158 NEXTI
6160 PRINTCHR$(147):GOTO6090
8000 REM AENDERUNGSTEIL
8010 PRINTCHR$(147)
8020 INPUT" NUMMER DER ZEICHENKETTE";N
8030 PRINT:PRINTT$(N-100*(Z-1))
8040 PRINT:INPUT" NEUE ZEICHENKETTE";T$(N-100*(Z-1)):PRINTCHR$(14
```

```
7):GOTO6100
9000 CLOSE2:CLOSE5:PRINTCHR$(147):END
10000 REM MATRIX LOESCHEN
10010 FORI=0TO99:T$(I)="":NEXTI:RETURN
```

Der Assembler

Der Assembler verlangt die Einhaltung einiger Konventionen. Er erwartet, daß in der Quelltextdatei Ansprungmarken, Befehle und Operanden getrennt voneinander als Einzelzeichenketten vorhanden sind. Dies läßt sich ganz einfach bewerkstelligen, indem man im Editor die Elemente einzeln eingibt. Darüberhinaus sind noch einige syntaxabhängige Sonderzeichen zu berücksichtigen, die zur Kennzeichnung der Adressierungsarten verwendet werden.

Die im Buch aufgeführten Beispielprogramme zeigen die Darstellung, die der Assembler ausdruckt. Neben dem Quelltext werden Adressen und Hex-Code angezeigt. Die mit einem Sternchen (*) versehenen Zeilen dienen lediglich als Erklärung für den Leser. Für den Assembler muß in den Editor nur der Quelltext eingegeben werden.

Wertzuweisung und Ansprungpunkte müssen als erstes Zeichen immer ein Gleichheitszeichen (=) besitzen. Sie dürfen eine Länge von acht Zeichen (einschließlich Gleichheitszeichen) nicht überschreiten.

Für Wertzuweisungen gilt folgende Regel:

```
                @Zahl              =Symbol

Beispiele:      @1000              =WERT1
                @$FF00             =WERT2
```

Befehls-Sonderzeichen für Adressierungsarten

Adressierungs-arten	an den Befehl anzuhängendes Zeichen	Beispiel	
		Mnemonik	Hex-Code
Immediate	#	CMP# 'A	C941
Absolute		JMP ;MARKE	4C????
Zero Page		INC. $30	E630
Indexed indirect	@X	LDA@X #10	A10A
Indirect indexed	@Y	SBC@Y #48	F130
Zero Page indexed,X	.X	ORA.X $10	1510
Zero Page indexed,Y	.Y	STX.Y #1	9601
Absolute indexed,X	X	STAX ;#256	9D0001
Absolute indexed,Y	Y	EORY ;$FF01	5901FF
Accumulator	A	ASLA	06
Indirect	@	JMP@ ;$4000	6C0040
Implied		INX	E8
Relativ		BNE +LOOP	D0??

Für die Verarbeitung der Operanden gelten folgende Regeln:

- Operanden, die aus einem Byte bestehen, erhalten immer ein Sonderzeichen vorangestellt, das kennzeichnet, wie sie zu interpretieren sind.

```
CMP#  .WERT       Symbol WERT (wurde vorher zugewiesen)
LDA.  #10         Dezimalzahl 10
AND.X $FF         Hexadezimalzahl $FF
LDA#  'A          ASCII-Wert 'A'
```

- Adressen (2 Byte) müssen immer einen Strichpunkt (;) vorangestellt bekommen.

```
LDA   ;#1000      Dezimalzahl 1000
STA   ;$A000      Hexadezimalzahl $A000
JMP   ;CLOSE      Symbol CLOSE
```

- Die Angabe des Sprungziels muß bei relativen Sprüngen durch ein Plus (+) gekennzeichnet sein.

```
              CMP#  $F0
              BEQ   +SPRUNG
              LDA   ;ALT
=SPRUNG       STA   ;NEU
```

Mit den Sonderzeichen '&' und '/' können Zeichenketten im Speicher abgelegt werden.

```
=NORMAL       &DAS IST TEXT

=EXTRA        /DAS IST TEXT
```

Die beiden Zeichenketten sind mit Ausnahme des letzten Byte identisch. Das vorangestellte Sonderzeichen '/' der 'Zeichenkette' =EXTRA bewirkt, daß beim Assemblieren zu diesem Byte der Wert 128 addiert wird. Das heißt, daß Bit 7 gesetzt wird.

```
1 POKE59468,12:PRINTCHR$(147)
2 REM DASM JANUAR 1984
10 REM
90 GOTO 62000
100 SIZ=60:DIMSY$(SIZ),AD(SIZ)
110 OP=56:DIMOP$(OP)
120 FORI=1TOOP:READOP$(I):NEXTI
900 PRINT" DURCHLAUF 1"
1000 REM DURCHLAUF 1
1004 IFFLG=1THEN2000
1005 INPUT#5,B$
1010 IFST<>0THENFLG=1
1020 C$=LEFT$(B$,1):L$=MID$(B$,2)
1030 IFC$="."ORC$="#"ORC$=$THENPC=PC+1:GOTO1000
1040 IFC$=";"THENPC=PC+2:GOTO1000
1050 IFC$="="THENGOSUB9000:PRINT B$:GOTO1000
1060 IFC$="@"THENGOSUB8100:PC=V:GOTO1000
1065 IFC$="&"ORC$="/"THENPC=PC+LEN(L$):GOTO1000
1070 PC=PC+1:GOTO1000
2000 REM DURCHLAUF 2
2004 PRINT" DURHLAUF 2"
2005 CLOSE5:OPEN5,8,5,"0:"+F$+",S,R"
2010 FLG=0
2020 PRINT
2100 IFFLG=1THEN3000
2102 INPUT#5,B$
2105 IFST<>0THENFLG=1
2110 IFASC(B$)>64ANDASC(B$)<=90THENA2$=B$:GOTO2120
2112 IFASC(B$)=61THENA1$=B$:GOTO2120
2115 A3$=B$
2120 C$=LEFT$(B$,1):L$=MID$(B$,2)
2130 IFC$="ANDF=1THENPRINTTAB(13);A3$;TAB(29);A1$:F=0:A1$="":A3$=
"":GOTO2100
2135 IFC$="="THENV=PC:GOSUB8300:PRINTV$;": ";:F=3:GOTO2100
2140 IFC$="@"THENGOSUB8100:PC=V:GOSUB8300:PRINTV$;:F=1:GOTO2100
2150 IFC$="#"ORC$="$"ORC$="'"THENL$="."+B$:GOSUB7000:GOTO2100
2160 IFC$=";"ORC$="."ORC$="+"THENL$=B$:GOSUB7000:GOTO2100
2170 GOSUB9500:L$=".$"+OP$:GOSUB7000
2180 GOTO2100
3000 PRINT:PRINT"ASSEMBLIERUNG BEENDET":CLOSE5
3010 END
7000 REM WERT ZUORDNEN
7010 C$=LEFT$(L$,1):L$=MID$(L$,2)
7020 IFC$<>"."ANDC$<>";"ANDC$<>"+"THENPRINT"FEHLERHAFTER OPERAND"
```

```
:RETURN
7030 GOSUB8200
7040 IFC$="+"THENV=V-(PC-1):IF(V<0ANDV>-129)ORV>127THENV=V+256
7050 IFC$<>";"AND(V<0ORV>255)THENPRINT"BYTEUEBERLAUF":V=0
7060 H=INT(V/256):V=V-256*H:C=C-1
7070 POKEPC,V:PC=PC+1
7080 GOSUB8300:VY$=VY$+RIGHT$(V$,2)
7085 IFC<0THENGOSUB8800:GOTO7900
7090 IFC>0THENRETURN
7100 IFTY$="-"ANDF=2THENPRINTVY$;TAB(13);A1$;TAB(23);A2$:GOTO7900
7110 IFTY$="-"THENV=PC-1:GOSUB8300:PRINTV$;": ";VY$;TAB(23);A2$:G
OTO7900
7115 IFTY$=";"THENGOTO7150
7120 IFF=2THENPRINTVY$;TAB(13);A1$;TAB(23);A2$;TAB(29);A3$:GOTO79
00
7130 V=PC-2:GOSUB8300:PRINTV$;":";VY$;TAB(23);A2$;TAB(29);A3$:GOT
O7900
7150 V=H:POKEPC,V:PC=PC+1
7160 GOSUB8300:VY$=VY$+RIGHT$(V$,2)
7170 IFF=2THENPRINTVY$;TAB(13);A1$;TAB(23);A2$;TAB(29);A3$:GOTO79
00
7180 V=PC-3:GOSUB8300:PRINTV$;": ";VY$;TAB(23);A2$;TAB(29);A3$
7900 VY$="":A1$="":A2$="":A3$="":F=0
7910 RETURN
8000 V=0:REM DEZIMALE WERTE
8010 T=ASC(V$)-48:IFT>9THENT=T-7
8020 V=16*V+T:V$=MID$(V$,2)
8030 IFV$<>""THEN8010
8040 RETURN
8100 REM WERTZUWEISUNG 1
8110 L1$=LEFT$(L$,1):L2$=MID$(L$,2)
8120 IFL1$="#"THENV=VAL(L2$):RETURN
8130 IFL1$="$"THENV$=L2$:GOSUB8000:RETURN
8135 IFL1$="'"THEN V=ASC(L2$):RETURN
8140 V=VAL(L$):RETURN
8200 REM WERTZUWEISUNG 2
8210 L1$=LEFT$(L$,1):L2$=MID$(L$,2)
8220 IFL1$="#"THENV=VAL(L2$):RETURN
8230 IFL1$="$"THENV$=L2$:GOSUB8000:RETURN
8235 IFL1$="'"THENV=ASC(L2$):RETURN
8240 GOSUB 9100:RETURN
8300 REM HEX UMWANDLUNG
8310 V$="":FORI=1TO4
8320 VX=INT(V/(16↑(4-I))):IFVX<>0THENV=V-(VX*16↑(4-I))
```

```
8330 GOSUB8500:NEXTI:RETURN
8500 VX$=CHR$(VX+48):IFVX>9THENVX$=CHR$(VX+55)
8510 V$=V$+VX$:RETURN
8600 X=0:IFF<>3THENV=PC:GOSUB8300:PRINTV$;": ";
8610 PRINTTAB(13);A1$;TAB(29);A3$
8620 FORJ=1TOLEN(L$)
8630 V=ASC(MID$(L$,J,1)):IFC$="/"ANDJ=LEN(L$)THEN V=V+128
8640 POKEPC,V:PC=PC+1:GOSUB8300:VY$=VY$+RIGHT$(V$,2):X=X+1
8650 IFX=3THENV=PC-3:GOSUB8300:PRINTV$;": ";VY$:VY$="":X=0
8660 NEXTJ
8670 IFX=0THENRETURN
8680 V=PC-X:GOSUB8300:PRINTV$;": ";VY$:RETURN
8800 IFF=3ANDC$<>";"THENPRINTVY$;TAB(13);A1$;TAB(29);A3$:RETURN
8810 IFC$<>";"THENV=PC-1:GOSUB8300:PRINTV$;": ";VY$;TAB(29);A3$:R
ETURN
8820 V=H:POKEPC,V:PC=PC+1
8830 GOSUB8300:VY$=VY$+RIGHT$(V$,2)
8840 IFF=3THENPRINTVY$;TAB(13);A1$;TAB(29);A3$:RETURN
8850 V=PC-2:GOSUB8300:PRINTV$;": ";VY$;TAB(29);A3$:RETURN
9000 REM ANSPRUNGMARKEN
9010 IFSY>SIZETHENPRINT"ZU VIELE ANSPRUNGMARKEN":RETURN
9020 SY$(SY)=L$:AD(SY)=PC:SY=SY+1:RETURN
9100 REM
9110 FORI=LEN(L$)TO1STEP-1
9120 T$=MID$(L$,I,1):IFT$="+"ORT$="-"THEN9140
9130 NEXTI:V=0:GOTO9160
9140 SV$=LEFT$(L$,I-1):L$=MID$(L$,I+1):GOSUB8100
9150 L$=SV$:IFT$="-"THENV=-V
9160 IFSY=0THEN9190
9170 FORI=0TOSY-1:IFSY$(I)=L$THENV=AD(I)+V:RETURN
9180 NEXTI
9190 PRINT"NICHT DEFINIERTE ANSPRUNGMARKE"
9200 GOSUB9000:V=PC+V:RETURN
9500 REM HEXKODEZUORDNUNG
9510 T$=LEFT$(B$,3):SF$=MID$(B$+"  ",4,2)
9520 FORI=1TOOP:OP$=OP$(I)
9530 IFLEFT$(OP$,3)=T$THEN9560
9540 NEXT I
9550 PRINT"UNGUELTIGER BEFEHL":OP$="0":TY$="-":RETURN
9560 FORI=4TOLEN(OP$)STEP5
9570 IFMID$(OP$,I,2)=SF$THEN9590
9580 NEXTI:GOTO9550
9590 TY$=MID$(OP$,I+4,1):OP$=MID$(OP$,I+2,2)
9595 IFF=3THENF=2
```

```
9600 IFTY$="-"THENC=1:RETURN
9610 C=2:RETURN
10000 DATA"ADC  6D;X 7D;Y 79;. 65..X75.@X61.@Y71.# 69."
10010 DATA"AND  2D;X 3D;Y 39;. 25..X35.@X21.@Y31.# 29."
10020 DATA"ASL  0E;X 1E;. 06..X16.A 0A-"
10030 DATA"BCC  90+","BCC  B0+","BEQ  F0+"
10040 DATA"BIT  2C;. 24.","BMI  30+","BNE  D0+","BPL  10+"
10050 DATA"BRK  00-","BVC  50+","BVS  70+"
10060 DATA"CLC  18-","CLD  D8-","CLI  58-","CLV  B8-"
10070 DATA"CMP  CD;X DD;Y D9;. C5..XD5.@XC1.@YD1.# C9."
10080 DATA"CPX  EC;. E4.# E0.","CPY  CC;. C4.# C0."
10090 DATA"DEC  CE;X DE;. C6..XD6.","DEX  CA-","DEY  88-"
10100 DATA"EOR  4D;X 5D;Y 59;. 45..X55.@X41.@Y51.# 49."
10110 DATA"INC  EE;X FE;. E6..XF6.","INX  E8-","INY  C8-"
10120 DATA"JMP  4C;@ 6C;","JSR  20;"
10130 DATA"LDA  AD;X BD;Y B9;. A5..XB5.@XA1.@YB1.# A9."
10140 DATA"LDX  AE;Y BE;. A6..YB6.# A2."
10150 DATA"LDY  AC;X BC;. A4..XB4.# A0."
10160 DATA"LSR  4E;X 5E;. 46..X56.A 4A-"
10170 DATA"NOP  EA-"
10180 DATA"ORA  OD;X 1D;Y 19;. 05..X15.@X01.@Y11.@ 09."
10190 DATA"PHA  48-","PHP  08-","PLA  68-","PLP  28-"
10200 DATA"ROL  2E;X 3E;. 26..X36.A 2A-"
10210 DATA"ROR  6E;X 7E;. 66..X76.A 6A-"
10220 DATA"RTI  40-","RTS  60-"
10230 DATA"SBC  ED;X FD;Y F9;. E5..XF5.@XE1.@YF1.# E9."
10240 DATA"SEC  38-","SED  F8-","SEI  78-"
10250 DATA"STA  8D;X 9D;Y 99;. 85..X95.@X81.@Y91."
10260 DATA"STX  8E;. 86..Y96.","STY  8C;. 84..X94."
10270 DATA"TAX  AA-","TAY  A8-","TSX  BA-","TXA  8A-","TXS  9A-",
"TYA  98-"
62000 PRINTCHR$(147)
62005 PRINT:PRINT"   COPYRIGHT (C) 1984";TAB(22);"BY M+T, HAAR"
62010 FORI=1TO10:PRINT"----";:NEXT
62015 PRINT:PRINT:PRINT:PRINTTAB(6)"EIN EINFACHER 6502 ASSEMBLER"
62020 PRINT:PRINT"      WEITER --> RETURN DRUECKEN"
62030 GETT$:IFT$=""THEN62030
62035 PRINT:PRINT:PRINT"      NAME DER QUELTEXTDATEI":PRINT:INPUT
"      ";F$
62040 PRINTCHR$(147):OPEN5,8,5,"0:"+F$;",S,R":GOTO100
```

Lieferbare Markt & Technik-Titel:

CP/M und WordStar Anwenderhandbuch
Best.-Nr. MT 310 DM 29,80*

Software-Auswahl leicht gemacht
Best.-Nr. MT 340 DM 58,—*

Hardware-Auswahl leicht gemacht
3. überarbeitete und aktualisierte Ausgabe 1984/85
Best.-Nr. MT 350 DM 58,—*

Personal Computer Lexikon
Best.-Nr. MT 390 DM 19,80*

Planen und kalkulieren mit VisiCalc
Best.-Nr. MT 450 DM 32,—*

Basic ohne Probleme: Bd. 1 Unterweisung
Best.-Nr. MT 480 DM 36,—*

Basic ohne Probleme: Bd. 2 Übungen
Best.-Nr. MT 490 DM 26,—*

Basic ohne Probleme: Bd. 3 Programmentwicklung und Datenverwaltung
Best.-Nr. MT 500 DM 44,—*

Basic ohne Probleme: Bd. 4 Allgemeine Dateiverwaltung am praktischen Beispiel
Best.-Nr. MT 514 DM 53,—*

Basic-Programme für CBM/VC-20-Computer
Best.-Nr. MT 501 DM 32,—*

Planen und kalkulieren mit Multiplan
Best.-Nr. MT 502 DM 58,—*

Der IBM-Personal Computer
Best.-Nr. MT 503 DM 53,—*

Datenkommunikation und Lokale Computer-Netzwerke
Best.-Nr. MT 504 DM 58,—*

Software richtig eingekauft
Best.-Nr. MT 505 DM 34,—*

Wörterbuch der Daten- und Telekommunikation
Best.-Nr. MT 506 DM 38,—*

Multiplan richtig eingesetzt
Best.-Nr. MT 507 DM 58,—*

Multiplan richtig eingesetzt — Beispiele auf Diskette (5¼", IBM-PC mit MS-DOS 2.0)
Best.-Nr. MT 623 DM 48,—*

Personal Computer — das intelligente Werkzeug für jedermann
Best.-Nr. MT 508 DM 53,—*

SuperCalc richtig eingesetzt
Best.-Nr. MT 511 DM 58,—*

SuperCalc richtig eingesetzt — Beispiele auf Diskette (5¼", IBM-PC mit MS-DOS 2.0)
Best.-Nr. MT 621 DM 48,—*

Programme und Tips für VC-20
Best.-Nr. MT 513 DM 38,—*

Das VC-20 Buch
Best.-Nr. MT 516 DM 49,—*

Das VC-20 Buch — Beispiele auf Kassette
Best.-Nr. MT 581 DM 19,90*

Das VC-20 Buch — Beispiele auf Diskette
Best.-Nr. MT 582 DM 29,90*

Ein-Chip-Mikrocomputer-Handbuch
Best.-Nr. MT 517 DM 58,—*

Die Btx-Fibel
Best.-Nr. MT 519 DM 29,80*

Das Datenbanksystem dBASE II
Best.-Nr. MT 524 DM 68,—*

Basic-80 und CP/M
Best.-Nr. MT 525 DM 48,—*

Einführung in Datenbanksysteme mit dBASE II
Best.-Nr. MT 526 DM 68,—*

Einführung in Datenbanksysteme mit dBASE II — Beispiele auf Diskette (5¼", IBM-PC mit MS-DOS 2.0)
Best.-Nr. MT 622 DM 48,—*

dBASE II richtig eingesetzt
Best.-Nr. MT 541 DM 68,—*

dBASE II richtig eingesetzt — Beispiele auf Diskette (5¼", IBM-PC mit MS-DOS 2.0))
Best.-Nr. MT 544 DM 48,—*

Einführung in C
Best.-Nr. MT 561 DM 69,—*

Mit Lotus 1-2-3 zur integrierten Problemlösung
Best.-Nr. MT 562 DM 68,—*

* inkl. MwSt. zuzügl. Versandkosten

Markt&Technik

Hans-Pinsel-Straße 2 · 8013 Haar bei München · Telefon 4613-220

Lieferbare Markt & Technik-Titel:

Basic-Dialekte im Vergleich Best.-Nr. MT 564	DM 32,—*
Das Commodore 64-Buch, Bd. 1: Leitfaden für Erstanwender — mit Assembler Best.-Nr. MT 591	DM 48,—*
Das Commodore 64-Buch, Bd. 1: Beispiele auf Diskette Best.-Nr. MT 592	DM 58,—*
Das Commodore 64-Buch, Bd. 2: Basic-Spiele Best.-Nr. MT 593	DM 38,—*
Das Commodore 64-Buch, Bd. 2: Beispiele auf Diskette Best.-Nr. MT 594	DM 58,—*
Das Commodore 64-Buch, Bd. 3: Leitfaden für Fortgeschrittene Best.-Nr. MT 595	DM 38,—*
Das Commodore 64-Buch, Bd. 3: Beispiele auf Diskette Best.-Nr. MT 596	DM 58,—*
Das Commodore 64-Buch, Bd. 4: Leitfaden für Programmierer — Assembler — Disassembler Best.-Nr. MT 597	DM 38,—*
Das Commodore 64-Buch, Bd. 4: Beispiele auf Diskette Best.-Nr. MT 598	DM 58,—*
Das Commodore 64-Buch, Bd. 5: Simon's Basic Best.-Nr. MT 599	DM 38,—*
Das Commodore 64-Buch, Bd. 5: Beispiele auf Diskette Best.-Nr. MT 600	DM 58,—*
Das Commodore 64-Buch, Bd. 6: Spiele Best.-Nr. MT 619	DM 38,—*
Das Commodore 64-Buch, Bd. 6: Beispiele auf Diskette Best.-Nr. MT 620	DM 58,—*
Computerspiele und Wissenswertes — Commodore 64 Best.-Nr. MT 601	DM 29,80*
Computerspiele und Wissenswertes — Commodore 64: Beispiele auf Diskette Best.-Nr. MT 602	DM 38,—*
Das große Spielebuch Commodore 64 Best.-Nr. MT 603	DM 29,80*
Das große Spielebuch Commodore 64: Beispiele auf Diskette Best.-Nr. MT 604	DM 38,—*
Software-Schnellkurs: CP/M Best.-Nr. MT 605	DM 37,—*
Software-Schnellkurs: MailMerge Best.-Nr. MT 606	DM 37,—*
Software-Schnellkurs: dBASE II Best.-Nr. MT 607	DM 37,—*
Software-Schnellkurs: SuperCalc Best.-Nr. MT 608	DM 37,—*
Software-Schnellkurs: WordStar Best.-Nr. MT 609	DM 37,—*
Software-Schnellkurs: Multiplan Best.-Nr. MT 610	DM 37,—*
Software-Schnellkurs: Lotus 1-2-3 Best.-Nr. MT 611	DM 37,—*
Software-Schnellkurs: CP/M 86 Best.-Nr. MT 615	DM 37,—*
Mehr als 32 Basic-Programme für den Commodore 64 Best.-Nr. MT 613	DM 49,—*
Mehr als 32 Basic-Programme für den Commodore 64: Beispiele auf Diskette Best.-Nr. MT 614	DM 48,—*
Mehr als 32 Basic-Programme für den IBM-PC Best.-Nr. MT 624	DM 68,—*
Mehr als 32 Basic-Programme für den IBM-PC: Beispiele auf Diskette (5¼" mit MS-DOS 2.0) Best.-Nr. MT 625	DM 58,—*
MS-DOS Best.-Nr. MT 616	DM 58,—*
Einführung in Forth Best.-Nr. 635	DM 58,—*
Lotus 1-2-3 richtig eingesetzt Best.-Nr. MT 637	DM 68,—*
Lotus 1-2-3 richtig eingesetzt Beispiele auf Diskette (5¼", IBM-PC mit MS-DOS 2.0) Best.-Nr. MT 638	DM 58,—*

* inkl. MwSt. zuzügl. Versandkosten

Markt&Technik

Hans-Pinsel-Straße 2 · 8013 Haar bei München · Telefon 4613-220

Computerspiele & Wissenswertes Commodore 64

Immer mehr Besitzer eines Commodore 64 beginnen für viele Anwendungsfälle Maschinenprogramme zu schreiben. Häufig reicht die Ausführungsgeschwindigkeit eines Basic-Programms nicht mehr aus, so daß die einzige Alternative in der Erstellung eines Assembler-Programms besteht. Dabei tauchen immer wieder die gleichen Probleme auf. Für den einen Drucker braucht man eine Centronics-Schnittstelle, für einen anderen ein serielles V.24-Interface. Manchmal benötigt man die exakte Tageszeit, oder man möchte gern den Befehlssatz des Basic erweitern. Schnelle binäre Arithmetik ist nur in der Maschinenprogrammebene möglich.

All dies und noch einiges mehr finden Sie in diesem Buch. Es ist als eine Assembler-Programmsammlung für den fortgeschrittenen Programmierer gedacht, der das Rad nicht jedesmal von neuem erfinden möchte. Der Einsatz der fertigen Problemlösungen wird durch einen als Listing im Buch enthaltenen Assembler unterstützt. Für alle diejenigen, denen das Eintippen zu mühsam ist, hält der Verlag eine Diskette mit allen im Buch gezeigten Programmen bereit.

Markt&Technik Verlag Aktiengesellschaft
ISBN 3-922120-62-8